

```

1: s*****
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: 19728384 V3.9.9.92 April 2014 To EKON/BASTA/JAX 18 2014
10: *****Now the Funclist*****
11: Funclist Function : 11963 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 7474 //6792 //6401 4752
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1231 //995 //
16: def head:max: maXbox7: 10.03.2014 19:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: -----
20: Funclist total Size all is: 20668! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 19510
22: ASize of EXE: 19728384 (16586240) (13511680) (13023744)
23: SHA1 Hash of maXbox 3.9.9.92: 61ECB8AF3FA0C72B6AA6DE8DA4A65E1C30F0EE6C
24:
25: -----
26: -----
27: //////////////////////////////////////
28:
29:
30: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
31: Function *****Now the Funclist*****
32: function GetResStringChecked(Ident: string; const Args: array of const): string
33: function ( Index : Longint) : Integer
34: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
35: function _CheckAutoResult( ResultCode : HRESULT ) : HRESULT
36: function _T(Name: tbtString): Variant;
37: function ABNFToText(const AText : String) : String
38: function Abs(e : Extended) : Extended;
39: function Ackermann( const A, B : Integer) : Integer
40: function AcquireLayoutLock : Boolean
41: function ActionByName( const AName : string) : TWebActionItem
42: function ACTIVEBUFFER : PCHAR
43: function Add : TAggregate
44: function Add : TCollectionItem
45: function Add : TColumn
46: function Add : TComboExItem
47: function Add : TCookie
48: function Add : TCoolBand
49: function Add : TFavoriteLinkItem
50: function Add : TFileTypeItem
51: function Add : THeaderSection
52: function Add : THTMLTableColumn
53: function Add : TIdEmailAddressItem
54: function Add : TIdMessagePart
55: function Add : TIdUserAccount
56: function Add : TListColumn
57: function Add : TListItem
58: function Add : TStatusPanel
59: function Add : TTaskDialogBaseButtonItem
60: function Add : TWebActionItem
61: function Add : TWorkArea
62: function Add( AClass : TClass) : Integer
63: function Add( AComponent : TComponent) : Integer
64: function Add( AItem, AData : Integer) : Integer
65: function Add( AItem, AData : Pointer) : Pointer
66: function Add( AItem, AData : TObject) : TObject
67: function Add( AObject : TObject) : Integer
68: function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
69: function Add( const S : WideString) : Integer
70: function Add( Image, Mask : TBitmap) : Integer
71: function Add( Index : Longint; const Text : string) : Longint
72: function Add( Sibling : TTreeNode; const S : string) : TTreeNode
73: function Add(const S: string): Integer
74: function Add(S: string): Integer;
75: function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
76: function ADDCHILD : TFIELDDEF
77: function AddChild( Index : Longint; const Text : string) : Longint
78: function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
79: function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
80: function AddChildObject( Index : Longint; const Text : string; const Data : Pointer) : Longint
81: function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
82: function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
83: function ADDFIELDDEF : TFIELDDEF
84: function AddFileExtIfNecessary( AFileName, AExt : string) : string
85: function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
86: function AddIcon( Image : TIcon) : Integer
87: function AddImage( Value : TCustomImageList; Index : Integer) : Integer
88: function ADDINDEXDEF : TINDEXDEF
89: function AddItem(const Caption:String;const ImageIdx,SelectImageIdx,OverlayImagIdx,
Indent:Int;Data:Ptr):TComboExItem

```

```

90: Function AddItem( Item : THeaderSection; Index : Integer) : THeaderSection
91: Function AddItem( Item : TListItem; Index : Integer) : TListItem
92: Function AddItem( Item : TStatusPanel; Index : Integer) : TStatusPanel
93: Function AddMapping( const FieldName : string) : Boolean
94: Function AddMasked( Image : TBitmap; MaskColor : TColor) : Integer
95: Function AddModuleClass( AClass : TComponentClass) : TComponent
96: Function AddModuleName( const AClass : string) : TComponent
97: Function AddNode(Node,Relative: TTreeNode;const S : string;Ptr:Pointer;Method:TNodeAttachMode):TTreeNode
98: Function AddObject( const S : WideString; AObject : TObject) : Integer
99: Function AddObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
100: Function AddObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
101: function AddObject(S:String;AObject:TObject):integer
102: Function AddObjectFirst( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
103: Function AddParameter : TParameter
104: Function 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 CallTerminateProc: Boolean
303: function CANFOCUS:BOOLEAN
304: Function CanLoad( const Ext : string) : Boolean
305: Function CanParse( AWebRequest : TWebRequest) : Boolean
306: Function CanSave( const Ext : string) : Boolean
307: Function CanStart( cChar : char) : boolean
308: Function CaptureScreen : TBitmap;
309: Function CaptureScreen1( Rec : TRect) : TBitmap;
310: Function CardinalToFourChar( ACardinal : LongWord) : string
311: Function CastSoapToNative(Info: PTypeInfo; const SoapData: WideString; NatData: Pointer; IsNull: Boolean): Boolean
312: Function CastSoapToVariant1( SoapInfo : PTypeInfo; const SoapData : WideString) : Variant;
313: Function Ceil( const X : Extended) : Integer
314: Function Ceil16( X : Integer) : Integer
315: Function Ceil4( X : Integer) : Integer
316: Function Ceil8( X : Integer) : Integer
317: Function Ceiling( const X : Extended) : Integer
318: Function CellRect( ACol, ARow : Longint) : TRect
319: Function CelsiusToFahrenheit( const AValue : Double) : Double
320: Function CenterPoint( const Rect : TRect) : TPoint
321: function CenterPoint(const Rect: TRect): TPoint
322: Function ChangeFileExt( FileName, Extension : string) : string
323: function ChangeFileExt(const FileName: string; const Extension: string): string
324: Function CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) : Boolean
325: Function CharInSet( const Ch : Char; const testSet : TSysCharSet) : Boolean
326: Function CharIsInEOF( const AString : string; ACharPos : Integer) : Boolean
327: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : string) : Boolean
328: Function CharLength( S : string; Index : Integer) : Integer
329: Function CharRange( const AMin, AMax : Char) : string
330: function CharSetToIdent(Charset: Longint; var Ident: string): Boolean
331: Function CharToBin(vChr: Char): string;
332: Function CharNext(lpsz: PChar): PChar; stdcall;
333: Function CharToByteIndex( S : string; Index : Integer) : Integer
334: function CharToByteIndex(const S: string; Index: Integer): Integer
335: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
336: function CharToByteLen(const S: string; MaxLen: Integer): Integer
337: Function CharToHex(const APrefix : string; const cc : Char) : shortstring;
338: function CharToHexStr(Value: char): string;
339: function CharToOem(ins, outs: PChar):boolean;
340: function CharToUnicode(Value: Char): string;
341: Function CheckMenuDropdown : Boolean
342: Function CheckMessages : longint
343: Function CheckOpen( Status : DBIResult) : Boolean
344: Function CheckPassword( const APassword : string) : Boolean
345: Function CheckResponse(const AResponse: SmallInt; const AAllowedResponses: array of SmallInt): SmallInt
346: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
347: function CheckSynchronize(Timeout: Integer): Boolean
348: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
349: function ChrA(const a: byte): char;
350: Function ClassIDToProgID(const ClassID: TGUID): string;
351: Function ClassNameIs(const Name: string): Boolean
352: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
353: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
354: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
355: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;
356: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;

```



```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

```

1310: Function GetTableNameFromQuery( const SQL : WideString) : WideString
1311: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1312: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1313: Function GetTasksList( const List : TStringList) : Boolean
1314: Function getTeamViewerID: string;
1315: Function GetTemplatesFolder : string
1316: Function GetText : PWideChar
1317: function GetText:PChar
1318: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1319: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1320: Function GetTextItem( const Value : string) : Longint
1321: function GETTEXTLEN:INTEGER
1322: Function GetThreadLocale: Longint; stdcall
1323: Function GetCurrentThreadID: LongWord; stdcall;
1324: Function GetTickCount : Cardinal
1325: Function 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(hwnd: HWND): HDC;
1356: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1357: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1358: Function GetWindowsComputerID : string
1359: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1360: Function GetWindowsFolder : string
1361: Function GetWindowsServicePackVersion : Integer
1362: Function GetWindowsServicePackVersionString : string
1363: Function GetWindowsSystemFolder : string
1364: Function GetWindowsTempFolder : string
1365: Function GetWindowsUserID : string
1366: Function GetWindowsVersion : TWindowsVersion
1367: Function GetWindowsVersionString : string
1368: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1369: Function GMTToLocalDateTime( S : string) : TDateTime
1370: Function GotoKey : Boolean
1371: Function GradToCycle( const Grads : Extended) : Extended
1372: Function GradToDeg( const Grads : Extended) : Extended
1373: Function GradToDeg( const Value : Extended) : Extended;
1374: Function GradToDeg1( const Value : Double) : Double;
1375: Function GradToDeg2( const Value : Single) : Single;
1376: Function GradToRad( const Grads : Extended) : Extended
1377: Function GradToRad( const Value : Extended) : Extended;
1378: Function GradToRad1( const Value : Double) : Double;
1379: Function GradToRad2( const Value : Single) : Single;
1380: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1381: Function GreenComponent( const Color32 : TColor32) : Integer
1382: Function GUIDToString(const GUID: TGUID): string)
1383: Function HandleAllocated : Boolean
1384: function HandleAllocated: Boolean;
1385: Function HandleRequest : Boolean
1386: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1387: Function HarmonicMean( const X : TDynFloatArray) : Float
1388: Function HasAsParent( Value : TTreeNode) : Boolean
1389: Function HASCHILDEFS : BOOLEAN
1390: Function HasCurValues : Boolean
1391: Function HasExtendCharacter( const s : UTF8String) : Boolean
1392: Function HasFormat( Format : Word) : Boolean
1393: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1394: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1395: Function HashValue(AStream: TStream): LongWord
1396: Function HashValue(AStream: TStream): Word
1397: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1398: Function HashValue1(AStream : TStream) : T4x4LongWordRecord

```



```

1399: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1400: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1401: Function HashValue16( const ASrc : string) : Word;
1402: Function HashValue16Stream( AStream : TStream) : Word;
1403: Function HashValue32( const ASrc : string) : LongWord;
1404: Function HashValue32Stream( AStream : TStream) : LongWord;
1405: Function HasMergeConflicts : Boolean
1406: Function hasMoreTokens : boolean
1407: Function HASPARENT : BOOLEAN
1408: function HasParent: Boolean
1409: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1410: Function HasUTF8BOM( S : TStream) : boolean;
1411: Function HasUTF8BOM1( S : AnsiString) : boolean;
1412: Function Haversine( X : Float) : Float
1413: Function Head( s : string; const subs : string; var tail : string) : string
1414: Function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1415: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1416: function HELPJUMP(JUMPID:STRING):BOOLEAN
1417: Function HeronianMean( const a, b : Float) : Float
1418: function HexStrToStr(Value: string): string;
1419: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1420: function HexToBin2(HexNum: string): string;
1421: Function HexToDouble( const Hex : string) : Double
1422: function HexToInt(hexnum: string): LongInt;
1423: function HexToStr(Value: string): string;
1424: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1425: function Hi(vdat: word): byte;
1426: function HiByte(W: Word): Byte)
1427: function High: Int64;
1428: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1429: function HINSTANCE: longword;
1430: function HiWord(l: DWORD): Word)
1431: function HMODULE: longword;
1432: Function HourOf( const AValue : TDateTime) : Word
1433: Function HourOfDay( const AValue : TDateTime) : Word
1434: Function HourOfMonth( const AValue : TDateTime) : Word
1435: Function HourOfTheWeek( const AValue : TDateTime) : Word
1436: Function HourOfTheYear( const AValue : TDateTime) : Word
1437: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1438: Function HourSpan( const ANow, AThen : TDateTime) : Double
1439: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1440: Function HTMLDecode( const AStr : String) : String
1441: Function HTMLEncode( const AStr : String) : String
1442: Function HTMLEscape( const Str : string) : string
1443: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1444: Function HTTPDecode( const AStr : String) : string
1445: Function HTTPEncode( const AStr : String) : string
1446: Function Hypot( const X, Y : Extended) : Extended
1447: Function 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 : TJclFileMappingView) : Integer
1513: Function INDEXOF( FIELD : TFIELD) : INTEGER
1514: Function IndexOf( ID : LCID) : Integer
1515: Function INDEXOF( ITEM : TMENUIITEM) : 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 : TMENUIITEM) : INTEGER
1560: Function INSERTNEWLINEBEFORE( AITEM : TMENUIITEM) : 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 ISEMPY : 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 ISVALIDDCHAR( 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 LoadFileAsString( const FileName : string) : string
1741: Function LoadFromFile( const FileName : string) : TBitmapLoader
1742: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1743: Function LoadPackage(const Name: string): HMODULE
1744: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1745: Function LoadStr( Ident : Integer) : string
1746: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1747: Function LoadWideStr( Ident : Integer) : WideString
1748: Function LOCATE(const KEYFIELDS: String; const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1749: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HRESULT
1750: Function LockServer( fLock : LongBool) : HRESULT
1751: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean

```



```

1752: Function Log( const X : Extended) : Extended
1753: Function Log10( const X : Extended) : Extended
1754: Function Log2( const X : Extended) : Extended
1755: function LogBase10(X: Float): Float;
1756: Function LogBase2(X: Float): Float;
1757: Function LogBaseN(Base, X: Float): Float;
1758: Function LogN( const Base, X : Extended) : Extended
1759: Function LogOffOS : Boolean
1760: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1761: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
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 MaskIntLiteralToChar( 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 : MMRResult)
1850: Procedure MidiInCheck( Code : MMRResult)
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, syncflag: integer ): boolean;
2039: Function Point( X, Y : Integer ) : TPoint
2040: function Point( X, Y : Integer ): TPoint
2041: Function PointAssign( const X, Y : Integer ) : TPoint
2042: Function PointDist( const P1, P2 : TPoint ) : Double;
2043: function PointDist( const P1,P2: TFloatPoint ): Double;
2044: Function PointDist1( const P1, P2 : TFloatPoint ) : Double;
2045: function PointDist2( const P1,P2: TPoint ): Double;
2046: Function PointEqual( const P1, P2 : TPoint ) : Boolean
2047: Function PointIsNull( const P : TPoint ) : Boolean
2048: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint ) : Double
2049: Function Poly( const X : Extended; const Coefficients : array of Double ) : Extended
2050: Function PortTCPIPisOpen( dwPort : Word; ipAddressStr : String ) : boolean;
2051: Function IsTCPPortOpen( dwPort : Word; ipAddressStr : String ) : boolean;
2052: Function Pop : Pointer
2053: Function Pop : TObject
2054: Function PopnStdDev( const Data : array of Double ) : Extended
2055: Function PopnVariance( const Data : array of Double ) : Extended
2056: Function PopulationVariance( const X : TDynFloatArray ) : Float
2057: function Pos( SubStr, S : AnyString ) : Longint;
2058: Function PosEqual( const Rect : TRect ) : Boolean
2059: Function PosEx( const SubStr, S : string; Offset : Integer ) : Integer
2060: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt ) : Integer
2061: Function PosInStrArray( const SearchStr:string; Contents:array of string; const CaseSensitive:Boolean ):Integer
2062: Function Post1( AURL : string; const ASource : TStrings ) : string;
2063: Function Post2( AURL : string; const ASource : TStream ) : string;
2064: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream ) : string;
2065: Function PostData( const UserData : WideString; const CheckSum : DWORD ) : Boolean
2066: Function PostData( const UserData : WideString; const CheckSum : integer ) : Boolean
2067: function PostMessage( hWnd: HWND; Msg: longword; wParam: longint; lParam: longint ) : Boolean;
2068: Function Power( const Base, Exponent : Extended ) : Extended
2069: Function PowerBig( aval, n:integer ): string;
2070: Function PowerIntJ( const X : Float; N : Integer ) : Float;
2071: Function PowerJ( const Base, Exponent : Float ) : Float;
2072: Function PowerOffOS : Boolean
2073: Function PreformatDateString( Ps : string ) : string
2074: Function PresentValue( const Rate:Extend; NPeriods:Int; const Payment,
    FutureVal:Extend; PaymentTime:TPaymentTime ):Extended
2075: Function PrimeFactors( N : Cardinal ) : TDynCardinalArray
2076: Function Printer : TPrinter
2077: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string ): string
2078: Function ProcessResponse : TIdHTTPWhatsNext
2079: Function ProduceContent : string
2080: Function ProduceContentFromStream( Stream : TStream ) : string
2081: Function ProduceContentFromString( const S : string ) : string
2082: Function ProgIDToClassID( const ProgID: string ): TGUID;
2083: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString ) : WideString
2084: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString ) : WideString
2085: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
    const ATitle : string; const AInitialDir : string; SaveDialog : Boolean ) : Boolean
2086: function PromptForFileName( var AFileName: string; const AFilter: string; const ADefaultExt: string; const
    ATitle: string; const AInitialDir: string; SaveDialog: Boolean ): Boolean
2087: Function PSScriptNeedFile( Sender:TObject; const OrginFileName:String; var FileName,Output:String ):Boolean
2088: Function PtInRect( const Rect : TRect; const P : TPoint ) : Boolean
2089: function PtInRect( const Rect: TRect; const P: TPoint ): Boolean
2090: Function Push( AItem : Pointer ) : Pointer
2091: Function Push( AObject : TObject ) : TObject
2092: Function Put1( AURL : string; const ASource : TStream ) : string;

```



```

2093: Function Pythagoras( const X, Y : Extended) : Extended
2094: Function queryDLLInterface( var queryList : TStringList) : TStringList
2095: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2096: Function QueryInterface(const IID: TGUID; out Obj): HRESULT, CdStdCall
2097: Function queryPerformanceCounter2(mse: int64): int64;
2098: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2099: //Function QueryPerformanceFrequency(mse: int64): boolean;
2100: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2101: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2102: Procedure QueryPerformanceCounter1(var aC: Int64);
2103: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2104: Function Quote( const ACommand : String) : SmallInt
2105: Function QuotedStr( S : string) : string
2106: Function RadToCycle( const Radians : Extended) : Extended
2107: Function RadToDeg( const Radians : Extended) : Extended
2108: Function RadToDeg( const Value : Extended) : Extended;
2109: Function RadToDeg1( const Value : Double) : Double;
2110: Function RadToDeg2( const Value : Single) : Single;
2111: Function RadToGrad( const Radians : Extended) : Extended
2112: Function RadToGrad( const Value : Extended) : Extended;
2113: Function RadToGrad1( const Value : Double) : Double;
2114: Function RadToGrad2( const Value : Single) : Single;
2115: Function RandG( Mean, StdDev : Extended) : Extended
2116: function Random(const ARange: Integer): Integer;
2117: function random2(a: integer): double
2118: function RandomE: Extended;
2119: function RandomF: Extended;
2120: Function RandomFrom( const AValues : array of string) : string;
2121: Function RandomRange( const AFrom, ATo : Integer) : Integer
2122: function randSeed: longint
2123: Function RawToDataColumn( ACol : Integer) : Integer
2124: Function Read : Char
2125: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HRESULT
2126: function Read(Buffer:String;Count:LongInt):LongInt
2127: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2128: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2129: Function ReadCardinal( const AConvert : boolean) : Cardinal
2130: Function ReadChar : Char
2131: Function ReadClient( var Buffer, Count : Integer) : Integer
2132: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2133: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2134: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2135: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
Boolean):Integer
2136: Function ReadInteger( const AConvert : boolean) : Integer
2137: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2138: Function ReadLn : string
2139: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2140: function ReadLn(question: string): string;
2141: Function ReadLnWait( AFailCount : Integer) : string
2142: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2143: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2144: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2145: Function ReadString( const ABytes : Integer) : string
2146: Function ReadString( const Section, Ident, Default : string) : string
2147: Function ReadString( Count : Integer) : string
2148: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2149: Function ReadTimeStampCounter : Int64
2150: Function RebootOS : Boolean
2151: Function Receive( ATimeOut : Integer) : TReplyStatus
2152: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2153: Function ReceiveLength : Integer
2154: Function ReceiveText : string
2155: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2156: Function ReceiveSerialText: string
2157: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2158: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
AMilliSec:Word):TDateTime
2159: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2160: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2161: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2162: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2163: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2164: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2165: Function RecodeTime( const AValue : TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2166: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2167: Function Reconcile( const Results : OleVariant) : Boolean
2168: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2169: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2170: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2171: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2172: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2173: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2174: Function RectCenter( const R : TRect) : TPoint
2175: Function RectEqual( const R1, R2 : TRect) : Boolean
2176: Function RectHeight( const R : TRect) : Integer
2177: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2178: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2179: Function RectIntersection( const R1, R2 : TRect) : TRect

```

```

2180: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2181: Function RectIsEmpty( const R : TRect) : Boolean
2182: Function RectIsNull( const R : TRect) : Boolean
2183: Function RectIsSquare( const R : TRect) : Boolean
2184: Function RectIsValid( const R : TRect) : Boolean
2185: Function RectsAreValid( R : array of TRect) : Boolean
2186: Function RectUnion( const R1, R2 : TRect) : TRect
2187: Function RectWidth( const R : TRect) : Integer
2188: Function RedComponent( const Color32 : TColor32) : Integer
2189: Function Refresh : Boolean
2190: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2191: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2192: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2193: Function RegisterConversionType(const AFam:TConvFam;const ADescr:string;const AFact:Double):TConvType
2194: Function RegistryRead(keyHandle: Longint; keyPath, myField: String) : string;
2195: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2196: Function ReleaseHandle : HBITMAP
2197: Function ReleaseHandle : HENHMETAFILE
2198: Function ReleaseHandle : HICON
2199: Function ReleasePalette : HPALETTE
2200: Function RemainderFloat( const X, Y : Float) : Float
2201: Function Remove( AClass : TClass) : Integer
2202: Function Remove( AComponent : TComponent) : Integer
2203: Function Remove( AItem : Integer) : Integer
2204: Function Remove( AItem : Pointer) : Pointer
2205: Function Remove( AItem : TObject) : TObject
2206: Function Remove( AObject : TObject) : Integer
2207: Function RemoveBackslash( const PathName : string) : string
2208: Function RemoveDF( aString : String) : String //removes thousand separator
2209: Function RemoveDir( Dir : string) : Boolean
2210: function RemoveDir(const Dir: string): Boolean)
2211: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2212: Function RemoveFileExt( const FileName : string) : string
2213: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2214: Function RenameFile( OldName, NewName : string) : Boolean
2215: function RenameFile(const OldName: string; const NewName: string): Boolean)
2216: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2217: Function ReplaceText( const AText, AFromText, AToText : string) : string
2218: Function Replicate(c : char;I : longInt) : String;
2219: Function Request : TWebRequest
2220: Function ResemblesText( const AText, AOther : string) : Boolean
2221: Function Reset : Boolean
2222: function Reset2(mypath: string):string;
2223: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2224: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2225: Function Response : TWebResponse
2226: Function ResumeSupported : Boolean
2227: Function RETHINKHOTKEYS : BOOLEAN
2228: Function RETHINKLINES : BOOLEAN
2229: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2230: Function RetrieveCurrentDir : string
2231: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2232: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2233: Function RetrieveMailBoxSize : integer
2234: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2235: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2236: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2237: Function ReturnMimeType( var MediaType, EncType : String) : Boolean
2238: Function ReverseBits( Value : Byte) : Byte;
2239: Function ReverseBits1( Value : Shortint) : Shortint;
2240: Function ReverseBits2( Value : Smallint) : Smallint;
2241: Function ReverseBits3( Value : Word) : Word;
2242: Function ReverseBits4( Value : Cardinal) : Cardinal;
2243: Function ReverseBits4( Value : Integer) : Integer;
2244: Function ReverseBits5( Value : Int64) : Int64;
2245: Function ReverseBytes( Value : Word) : Word;
2246: Function ReverseBytes1( Value : Smallint) : Smallint;
2247: Function ReverseBytes2( Value : Integer) : Integer;
2248: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2249: Function ReverseBytes4( Value : Int64) : Int64;
2250: Function ReverseString( const AText : string) : string
2251: Function ReverseDNSLookup(const IPAddr: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 : TFatchOptions) : 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: Boolean; 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_ReplFirst( const cAT, cStr, cRepl : string) : string
2297: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2298: Function S_RTrim( const cStr : string) : string
2299: Function S_StripTrimCopy( const cStr : string; iPos, iLen : integer) : string
2300: //Type TS_ShellExecuteCmd = (seCmdOpen, seCmdPrint, seCmdExplore);
2301: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2302: Function S_Space( const iLen : integer) : String
2303: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2304: Function S_StrBlanksCutTooLong( const cStr : string; const iLen : integer) : string
2305: Function S_StrCRC32( const Text : string) : LongWord
2306: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2307: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2308: Function S_StringToUTF_8( const AString : string) : string
2309: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2310: function S_StrToReal( const cStr : string; var R: Double): Boolean
2311: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2312: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2313: Function S_UTF_8ToString( const AString : string) : string
2314: Function S_WBox( const AText : string) : integer
2315: Function SameDate( const A, B : TDateTime) : Boolean
2316: function SameDate( const A, B : TDateTime): Boolean;
2317: Function SameDateTime( const A, B : TDateTime) : Boolean
2318: function SameDateTime( const A, B : TDateTime): Boolean;
2319: Function SameFileName( S1, S2 : string) : Boolean
2320: Function SameText( S1, S2 : string) : Boolean
2321: function SameText( const S1 : string; const S2 : string): Boolean)
2322: Function SameTime( const A, B : TDateTime) : Boolean
2323: function SameTime( const A, B : TDateTime): Boolean;
2324: function SameValue( const A, B: Extended; Epsilon: Extended): Boolean //overload;
2325: function SameValue1( const A, B: Double; Epsilon: Double): Boolean //overload;
2326: function SameValue2( const A, B: Single; Epsilon: Single): Boolean //overload;
2327: Function SampleVariance( const X : TDynFloatArray) : Float
2328: Function Sar( const Value : ShortInt; const Count : TBitRange) : ShortInt;
2329: Function Sar1( const Value : SmallInt; const Count : TBitRange) : SmallInt;
2330: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2331: Function SaveToFile( const AFileName : TFileName) : Boolean
2332: Function SaveAsExcelFile( AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2333: Function SaveAsExcel( aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2334: Function ScanF( const aformat: String; const args: array of const): string;
2335: Function SCREENTOCLIENT( POINT: TPOINT): TPOINT
2336: Function SearchBuf( Buf: PChar; BufLen: Integer; SelStart, SelLength: Integer; SearchString: String; Options:
TStringSearchOptions): PChar
2337: Function SearchBuf2( Buf: String; SelStart, SelLength: Integer; SearchString:
String; Options: TStringSearchOptions): Integer;
2338: function SearchRecAttr: integer;
2339: function SearchRecExcludeAttr: integer;
2340: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2341: function SearchRecName: string;
2342: function SearchRecSize: integer;
2343: function SearchRecTime: integer;
2344: Function Sec( const X : Extended) : Extended
2345: Function Secant( const X : Extended) : Extended
2346: Function SecH( const X : Extended) : Extended
2347: Function SecondOf( const AValue : TDateTime) : Word
2348: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2349: Function SecondOfTheHour( const AValue : TDateTime) : Word
2350: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2351: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2352: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2353: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2354: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64

```

```

2355: Function SecondSpan( const ANow, 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 SendMessage(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 SetLastErrorProc( ErrorProc : TSocketErrorProc ) : TSocketErrorProc
2399: Function SetFileCreation( const FileName : string; const DateTime : TDateTime ) : Boolean
2400: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime ) : Boolean
2401: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime ) : Boolean
2402: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer ) : Boolean
2403: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2404: Function SetLocalTime( Value : TDateTime ) : boolean
2405: Function SetPrecisionTolerance( NewTolerance : Float ) : Float
2406: Function SetPrinter( NewPrinter : TPrinter ) : TPrinter
2407: Function SetRGBValue( const Red, Green, Blue : Byte ) : TColor
2408: Function SetSequence( S, Localizar, Substituir : shortstring ) : shortstring
2409: Function SetSize( libNewSize : Longint ) : HRESULT
2410: Function SetUserObjectFullAccess( hUserObject : THandle ) : Boolean
2411: Function Sgn( const X : Extended ) : Integer
2412: function SHA1(const fileName: string): string;
2413: function SHA256(astr: string; amode: char): string)
2414: function SHA512(astr: string; amode: char): string)
2415: Function ShareMemoryManager : Boolean
2416: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2417: function Shellexecute2(hWnd: HWND; const FileName: string):integer; stdcall;
2418: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2419: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE ) : TSHORTCUT
2420: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT ) : String
2421: function ShortDateFormat: string;
2422: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
RTL:Bool;EllipsisWidth:Int):WideString
2423: function ShortTimeFormat: string;
2424: function SHOWMODAL:INTEGER
2425: function ShowWindow(C1: HWND; C2: integer): boolean;
2426: procedure ShowMemory //in Dialog
2427: function ShowMemory2: string;
2428: Function ShutDownOS : Boolean
2429: Function Signe( const X, Y : Extended ) : Extended
2430: Function Sign( const X : Extended ) : Integer
2431: Function Sin(e : Extended) : Extended;
2432: Function sinc( const x : Double ) : Double
2433: Function SinJ( X : Float ) : Float
2434: Function Size( const AFileName : String ) : Integer
2435: function SizeOf: Longint;
2436: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle ) : Integer
2437: function SlashSep(const Path, S: String): String
2438: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer ) : Extended
2439: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL ) : DWORD
2440: Function SmallPoint(X, Y: Integer): TSmallPoint)
2441: Function Soundex( const AText : string; ALength : TSoundexLength ) : string

```



```

2442: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2443: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2444: Function SoundexProc( const AText, AOther : string) : Boolean
2445: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2446: Function SoundexWord( const AText : string) : Word
2447: Function SourcePos : Longint
2448: function SourcePos:LongInt
2449: Function Split0( Str : string; const substr : string) : TStringList
2450: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2451: Function SQLRequiresParams( const SQL : WideString) : Boolean
2452: Function Sqr(e : Extended) : Extended;
2453: Function Sqrt(e : Extended) : Extended;
2454: Function StartIP : String
2455: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2456: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2457: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2458: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2459: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2460: Function StartOfAYear( const AYear : Word) : TDateTime
2461: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2462: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2463: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2464: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2465: Function StartsStr( const ASubText, AText : string) : Boolean
2466: Function StartsText( const ASubText, AText : string) : Boolean
2467: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2468: Function StartsWith( const str : string; const sub : string) : Boolean
2469: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2470: Function StatusString( StatusCode : Integer) : string
2471: Function StdDev( const Data : array of Double) : Extended
2472: Function Stop : Float
2473: Function StopCount( var Counter : TJclCounter) : Float
2474: Function StoreColumns : Boolean
2475: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2476: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2477: Function StrAlloc( Size : Cardinal) : PChar
2478: function StrAlloc(Size: Cardinal): PChar)
2479: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2480: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2481: Function StrBufSize( Str : PChar) : Cardinal
2482: function StrBufSize(const Str: PChar): Cardinal)
2483: Function StrByteType( Str : PChar; Index : Cardinal) : TMBcsByteType
2484: function StrByteType(Str: PChar; Index: Cardinal): TMBcsByteType)
2485: Function StrCat( Dest : PChar; Source : PChar) : PChar
2486: function StrCat(Dest: PChar; const Source: PChar): PChar)
2487: Function StrCharLength( Str : PChar) : Integer
2488: Function StrComp( Str1, Str2 : PChar) : Integer
2489: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2490: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2491: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2492: Function Stream_to_AnsiString( Source : TStream) : ansistring
2493: Function Stream_to_Base64( Source : TStream) : ansistring
2494: Function Stream_to_decimalbytes( Source : TStream) : string
2495: Function Stream2WideString( oStream : TStream) : WideString
2496: Function StreamtoAnsiString( Source : TStream) : ansistring
2497: Function StreamToByte( Source : TStream) : string
2498: Function StreamToDecimalbytes( Source : TStream) : string
2499: Function StreamtoOrd( Source : TStream) : string
2500: Function StreamToString( Source : TStream) : string
2501: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2502: Function StrEmpty( const sString : string) : boolean
2503: Function StrEnd( Str : PChar) : PChar
2504: function StrEnd(const Str: PChar): PChar)
2505: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2506: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2507: Function StrGet(var S : String; I : Integer) : Char;
2508: Function StrGet2(S : String; I : Integer) : Char;
2509: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2510: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2511: Function StrHtmlDecode( const AStr : String) : String
2512: Function StrHtmlEncode( const AStr : String) : String
2513: Function StrToBytes(const Value: String): TBytes;
2514: Function StrIComp( Str1, Str2 : PChar) : Integer
2515: Function StringOfChar(c : char;I : longInt) : String;
2516: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2517: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2518: Function StringRefCount(const s: String): integer;
2519: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2520: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2521: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2522: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2523: Function StringToBoolean( const Ps : string) : Boolean
2524: function StringToColor(const S: string): TColor)
2525: function StringToCursor(const S: string): TCursor;
2526: function StringToGUID(const S: string): TGUID)
2527: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2528: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2529: Function StringWidth( S : string) : Integer
2530: Function StrInternetToDateTime( Value : string) : TDateTime

```

```

2531: Function StrIsDateTime( const Ps : string ) : Boolean
2532: Function StrIsFloatMoney( const Ps : string ) : Boolean
2533: Function StrIsInteger( const S : string ) : Boolean
2534: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal ) : PChar
2535: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal ) : Integer
2536: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal ) : PChar
2537: Function StrLen( Str : PChar ) : Cardinal
2538: Function StrLen(const Str: PChar): Cardinal
2539: Function StrLessPrefix( const sString : string; const sPrefix : string ) : string
2540: Function StrLessSuffix( const sString : string; const sSuffix : string ) : string
2541: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal ) : Integer
2542: Function StrLower( Str : PChar ) : PChar
2543: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal ) : PChar
2544: Function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar
2545: Function StrNew( Str : PChar ) : PChar
2546: Function StrNew(const Str: PChar): PChar
2547: Function StrNextChar( Str : PChar ) : PChar
2548: Function StrPad( const sString : string; const sPad : string; const iLength : integer ) : string
2549: Function StrParse( var sString : string; const sDelimiters : string ) : string;
2550: Function 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( const Time : 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,AMonth,AMonth,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: Function TVariantRelationship, '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2847: Function VarSameValue( const A, B : Variant) : Boolean
2848: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2849: Function VarIsEmptyParam( const V : Variant) : Boolean
2850: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2851: Function VarIsError1( const V : Variant) : Boolean;
2852: Function VarAsError( AResult : HRESULT) : Variant
2853: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2854: Function VarIsArray( const A : Variant) : Boolean;
2855: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2856: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2857: Function VarArrayOf( const Values : array of Variant) : Variant
2858: Function VarArrayRef( const A : Variant) : Variant
2859: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2860: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2861: Function VarArrayDimCount( const A : Variant) : Integer
2862: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2863: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2864: Function VarArrayLock( const A : Variant) : __Pointer
2865: Procedure VarArrayUnlock( const A : Variant)
2866: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2867: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2868: Procedure DynArrayToVariant( var V : Variant; const DynArray : __Pointer; TypeInfo : __Pointer)
2869: Procedure DynArrayFromVariant( var DynArray : __Pointer; const V : Variant; TypeInfo : __Pointer)
2870: Function Unassigned : Variant
2871: Function Null : Variant
2872: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2873: Function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2874: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2875: Function VectorDot(const V1,V2: TFloatPoint): Double;
2876: Function VectorLengthSqr( const V : TFloatPoint) : Double

```

```

2877: function VectorLengthSqr(const V: TFloatPoint): Double;
2878: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint
2879: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2880: function VectorSubtract(const V1, V2: TFloatPoint): TFloatPoint
2881: function VectorSubtract(const V1, V2: TFloatPoint): TFloatPoint;
2882: function Verify( AUserName: String): String
2883: function Versine( X: Float): Float
2884: function VersionCheck: boolean;
2885: function VersionLanguageId(const LangIdRec: TLangIdRec): string
2886: function VersionLanguageName(const LangId: Word): string
2887: function VersionResourceAvailable(const FileName: string): Boolean
2888: function Visible: Boolean
2889: function VolumeID(DriveChar: Char): string
2890: function WaitFor(const AString: string): string
2891: function WaitFor(const Timeout: Cardinal): TJclWaitResult
2892: function WaitFor1: TWaitResult;
2893: function WaitForData( Timeout: Longint): Boolean
2894: function WebColorNameToColor( WebColorName: string): TColor
2895: function WebColorStrToColor( WebColor: string): TColor
2896: function WebColorToRGB( WebColor: Integer): Integer
2897: function wGet(aURL, afile: string): boolean;
2898: function wGet2(aURL, afile: string): boolean; //without file open
2899: function WebGet(aURL, afile: string): boolean;
2900: function WebExists: boolean; //alias to isinternet
2901: function WeekOf(const AValue: TDateTime): Word
2902: function WeekOfTheMonth(const AValue: TDateTime): Word;
2903: function WeekOfTheMonth1(const AValue: TDateTime; var AYear, AMonth: Word): Word;
2904: function WeekOfTheYear(const AValue: TDateTime): Word;
2905: function WeekOfTheYear1(const AValue: TDateTime; var AYear: Word): Word;
2906: function WeeksBetween(const ANow, AThen: TDateTime): Integer
2907: function WeeksInAYear(const AYear: Word): Word
2908: function WeeksInYear(const AValue: TDateTime): Word
2909: function WeekSpan(const ANow, AThen: TDateTime): Double
2910: function WideAdjustLineBreaks(const S: WideString; Style: TTextLineBreakStyle): WideString
2911: function WideCat(const x, y: WideString): WideString
2912: function WideCompareStr( S1, S2: WideString): Integer
2913: function WideCompareStr(const S1: WideString; const S2: WideString): Integer
2914: function WideCompareText( S1, S2: WideString): Integer
2915: function WideCompareText(const S1: WideString; const S2: WideString): Integer
2916: function WideCopy(const src: WideString; index, count: Integer): WideString
2917: function WideDequotedStr(const S: WideString; AQuote: WideChar): WideString
2918: function WideEqual(const x, y: WideString): Boolean
2919: function WideFormat(const Format: WideString; const Args: array of const): WideString
2920: function WideGreater(const x, y: WideString): Boolean
2921: function WideLength(const src: WideString): Integer
2922: function WideLess(const x, y: WideString): Boolean
2923: function WideLowerCase( S: WideString): WideString
2924: function WideLowerCase(const S: WideString): WideString
2925: function WidePos(const src, sub: WideString): Integer
2926: function WideQuotedStr(const S: WideString; Quote: WideChar): WideString
2927: function WideReplaceStr(const AText, AFromText, AToText: WideString): WideString
2928: function WideReplaceText(const AText, AFromText, AToText: WideString): WideString
2929: function WideSameStr( S1, S2: WideString): Boolean
2930: function WideSameStr(const S1: WideString; const S2: WideString): Boolean
2931: function WideSameText( S1, S2: WideString): Boolean
2932: function WideSameText(const S1: WideString; const S2: WideString): Boolean
2933: function WideStringReplace(const S, OldPattern, NewPattern: WideString; Flags: TReplaceFlags): WideString
2934: function WideStringToUCS4String(const S: WideString): UCS4String
2935: function WideUpperCase( S: WideString): WideString
2936: function Win32BackupFile(const FileName: string; Move: Boolean): Boolean
2937: function Win32Check(RetVal: boolean): boolean
2938: function Win32DeleteFile(const FileName: string; MoveToRecycleBin: Boolean): Boolean
2939: function Win32RestoreFile(const FileName: string): Boolean
2940: function Win32Type: TIdWin32Type
2941: function WinColor(const Color32: TColor32): TColor
2942: function winexec(FileName: pchar; showCmd: integer): integer;
2943: function WinExec32(const Cmd: string; const CmdShow: Integer): Boolean
2944: function WinExec32AndWait(const Cmd: string; const CmdShow: Integer): Cardinal
2945: function WithinPastDays(const ANow, AThen: TDateTime; const ADays: Integer): Boolean
2946: function WithinPastHours(const ANow, AThen: TDateTime; const AHours: Integer): Boolean
2947: function WithinPastMilliseconds(const ANow, AThen: TDateTime; const AMilliseconds: Integer): Boolean
2948: function WithinPastMinutes(const ANow, AThen: TDateTime; const AMinutes: Integer): Boolean
2949: function WithinPastMonths(const ANow, AThen: TDateTime; const AMonths: Integer): Boolean
2950: function WithinPastSeconds(const ANow, AThen: TDateTime; const ASeconds: Integer): Boolean
2951: function WithinPastWeeks(const ANow, AThen: TDateTime; const AWeeks: Integer): Boolean
2952: function WithinPastYears(const ANow, AThen: TDateTime; const AYears: Integer): Boolean
2953: function WNetAddConnection(lpRemoteName, lpPassword, lpLocalName: PChar): DWORD
2954: function WordToStr(const Value: Word): String
2955: function WordGridFormatIdentToInt(const Ident: string; var Value: Longint): Boolean
2956: function 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, ATen : TDateTime ) : Integer
2978: Function YearSpan( const ANow, ATen : 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: ($IFDEF WIN32)
3019: procedure ImageListDrawDisabled( Images: TImageList; Canvas: TCanvas;
  X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean );
3020: ($SENDIF)
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: ($IFDEF WIN32)
3026: function CreateIconFromBitmap( Bitmap: TBitmap; TransparentColor: TColor ): TIcon;
3027: ($SENDIF)
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 )
3048: Procedure CenterWindow( Wnd : HWND )
3049: Procedure ShadeRect( DC : HDC; const Rect : TRect )
3050: Procedure KillMessage( Wnd : HWND; Msg : Cardinal )

```

```

3053: Function DialogsToPixelsX( Dlgs : Word) : Word
3054: Function DialogsToPixelsY( Dlgs : Word) : Word
3055: Function PixelsToDialogsX( Pixs : Word) : Word
3056: Function PixelsToDialogsY( Pixs : Word) : Word
3057: { $IFDEF WIN32 }
3058: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3059: function MakeVariant(const Values: array of Variant): Variant;
3060: { $ENDIF }
3061: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3062: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3063: function MsgDlg(const Msg: string; AType: TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3064: { $IFDEF CBUILDER }
3065: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3066: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3067: { $ELSE }
3068: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3069: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3070: { $ENDIF CBUILDER }
3071: function IsForegroundTask: Boolean;
3072: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3073: function GetAveCharSize(Canvas: TCanvas): TPoint;
3074: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3075: procedure FreeUnusedOle;
3076: procedure Beep;
3077: function GetWindowsVersionJ: string;
3078: function LoadDLL(const LibName: string): THandle;
3079: function RegisterServer(const ModuleName: string): Boolean;
3080: { $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, 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 : PLOGFONT) : Boolean
3124: Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3125: Function Imm32IsIME( hK1 : longword) : Boolean
3126: Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3127: Procedure DragDone( Drop : Boolean)
3128:
3129:
3130: //*****added from jvjvclutils
3131: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3132: function ReplaceComponentReference(This, NewReference: TComponent; var VarReference: TComponent): Boolean;
3133: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3134: function IsPositiveResult(Value: TModalResult): Boolean;
3135: function IsNegativeResult(Value: TModalResult): Boolean;
3136: function IsAbortResult(const Value: TModalResult): Boolean;
3137: function StripAllFromResult(const Value: TModalResult): TModalResult;
3138: // returns either BrightColor or DarkColor depending on the luminance of AColor
3139: // This function gives the same result (AFAIK) as the function used in Windows to

```



```

3140: // calculate the desktop icon text color based on the desktop background color
3141: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3142: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3143:
3144: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3145:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3146:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3147:   var LinkName: string; Scale: Integer = 100); overload;
3148: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3149:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3150:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3151:   var LinkName: string; Scale: Integer = 100); overload;
3152: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3153:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3154: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3155:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3156:   Scale: Integer = 100): string;
3157: function HTMLPlainText(const Text: string): string;
3158: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3159:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3160: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3161:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3162: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3163: function HTMLPrepareText(const Text: string): string;
3164:
3165: ***** uPSI_JvAppUtils;
3166: function GetDefaultSection( Component : TComponent ) : string
3167: procedure GetDefaultIniData( Control: TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3168: procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3169: function GetDefaultIniName : string
3170:   //'OnGetDefaultIniName', 'TOnGetDefaultIniName').SetString();
3171: function GetDefaultIniRegKey : string
3172: function FindForm( FormClass : TFormClass ) : TForm
3173: function FindShowForm( FormClass : TFormClass; const Caption : string ) : TForm
3174: function ShowDialog( FormClass : TFormClass ) : Boolean
3175: //function InstantiateForm( FormClass : TFormClass; var Reference ) : TForm
3176: procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3177: procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3178: procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3179: procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3180: procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3181: function GetUniqueFileNameInDir( const Path, FileNameMask : string ) : string
3182: function StrToIniStr( const Str : string ) : string
3183: function IniStrToStr( const Str : string ) : string
3184: function IniReadString( IniFile : 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,erNullVariantErrorerInterfaceNotSupportedError);
3236: (*-----*)
3237: procedure SRegister_JclIniFiles(Cli: 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 SRegister_JclDateTime(Cli: 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 JBasterSunday( 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 StrFillChar( 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 StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3501: Procedure StrToStrings( 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}
3660: function CurrencyToStr(const Cur: currency): string;
3661: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3662: 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: { GetTempFileName returns temporary file name on drive, there FileName is placed }
3680: function GetTempFileName(const FileName: string): string;
3681: { GetTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3682: function GetTempFileNameExt(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: string; 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: string; 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 already 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: Boolean; const FileName: string): Boolean;
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: { RTextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3811: procedure RTextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3812: { RTextOutEx same with RTextOut function, but can calculate needed height for correct output }
3813: function RTextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: string; const CalcHeight: Boolean): Integer;
3814: { RTextCalcHeight calculate needed height to correct output, using RTextOut or RTextOutEx functions }
3815: function RTextCalcHeight(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 width 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, equal 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: { RBTtag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3836: function RBTtag(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(Parent: 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 TObjct }
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: TObjct; const PropName: string): TTypeKind;
3870: function GetPropStr(Obj: TObjct; const PropName: string): string;
3871: function GetPropOrd(Obj: TObjct; const PropName: string): Integer;
3872: function GetPropMethod(Obj: TObjct; const PropName: string): TMethod;
3873: procedure PrepareIniSection(SS: TStrings);
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:TObjct;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
3884:   Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3885:   Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
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 SRegister_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; ADelined, ASupress3D: 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 LoadComponentFromFile(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 VarSQLTimeStampCreate1( 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 : WideString; 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; and &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: {SENDIF 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="jan1 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="jan1 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: { MoveWideChar 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; {$IFDEF SUPPORTS_INLINE}
4529: {$ENDIF ~COMPILER12_UP}
4530: function CharInSetW(const Ch: WideChar; const SetOfChar: TSysCharSet): Boolean; inline; {$IFDEF
  SUPPORTS_INLINE}
4531:

```



```

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; {$ENDIF SUPPORTS_INLINE}
4541: TPixelFormat, '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4542: TMappingMethod, '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4543: function GetBitmapPixelFormat( Bitmap : TBitmap ) : TPixelFormat
4544: function GetPalettePixelFormat( Bitmap : TBitmap ) : TPixelFormat
4545: procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4546: function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4547: procedure GrayscaleBitmap( Bitmap : TBitmap)
4548: function BitmapToMemory( Bitmap : TBitmap; Colors : Integer ) : TStream
4549: procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4550: function ScreenPixelFormat : TPixelFormat
4551: function ScreenColorCount : Integer
4552: procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4553: function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean ) : TPoint
4554: // SIRegister_TJvGradient(CL);
4555:
4556: {***** files routines}
4557: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4558: const
4559:   {$IFDEF MSWINDOWS}
4560:   DefaultCaseSensitivity = False;
4561:   {$ENDIF MSWINDOWS}
4562:   {$IFDEF UNIX}
4563:   DefaultCaseSensitivity = True;
4564:   {$ENDIF UNIX}
4565: { GenTempFileName returns temporary file name on
4566: drive, there FileName is placed }
4567: function GenTempFileName(FileName: string): string;
4568: { GenTempFileNameExt same to previous function, but
4569: returning filename has given extension, FileExt }
4570: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4571: { ClearDir clears folder Dir }
4572: function ClearDir(const Dir: string): Boolean;
4573: { DeleteDir clears and than delete folder Dir }
4574: function DeleteDir(const Dir: string): Boolean;
4575: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4576: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4577: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4578: Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4579: function FileEquMasks(FileName, Masks: TFileName;
4580: CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4581: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4582: {$IFDEF MSWINDOWS}
4583: { LZFileExpand expand file, FileSource,
4584: into FileDest. Given file must be compressed, using MS Compress program }
4585: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4586: {$ENDIF MSWINDOWS}
4587: { FileGetInfo fills SearchRec record for specified file attributes}
4588: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4589: { HasSubFolder returns True, if folder APath contains other folders }
4590: function HasSubFolder(APath: TFileName): Boolean;
4591: { IsEmptyFolder returns True, if there are no files or
4592: folders in given folder, APath}
4593: function IsEmptyFolder(APath: TFileName): Boolean;
4594: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4595: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4596: { AddPath returns FileName with Path, if FileName not contain any path }
4597: function AddPath(const FileName, Path: TFileName): TFileName;
4598: function AddPaths(const PathList, Path: string): string;
4599: function ParentPath(const Path: TFileName): TFileName;
4600: function FindInPath(const FileName, PathList: string): TFileName;
4601: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4602: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4603: { HasParam returns True, if program running with specified parameter, Param }
4604: function HasParam(const Param: string): Boolean;
4605: function HasSwitch(const Param: string): Boolean;
4606: function Switch(const Param: string): string;
4607: { ExePath returns ExtractFilePath(ParamStr(0)) }
4608: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4609: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4610: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4611: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4612: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4613: {**** Graphic routines }
4614: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4615: 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 Phrases with the same Index. }
4643: function ReplaceAllStrings(const S: string; Words, Phrases: 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, Phrases, with the
   same Index, and then update NewSelStart variable }
4646: function ReplaceStrings(const S: string; PosBeg, Len: Integer; Words, Phrases: TStrings; var NewSelStart: Integer): 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 adds 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 beginning 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);
4699: {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 TObjct }
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: TObjct; const PropName: string): TTypeKind;
4711: function GetPropStr(Obj: TObjct; const PropName: string): string;
4712: function GetPropOrd(Obj: TObjct; const PropName: string): Integer;
4713: function GetPropMethod(Obj: TObjct; const PropName: string): TMethod;
4714: procedure PrepareIniSection(Ss: TStringList);
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(Dat1, Date2: TDateTime; var Days, Months, Years: Word);
4755: function MonthsBetween(Dat1, Date2: TDateTime): Double;
4756: function DaysInPeriod(Dat1, Date2: TDateTime): Longint;
4757: { Count days between Dat1 and Date2 + 1, so if Dat1 = Date2 result = 1 }
4758: function DaysBetween(Dat1, Date2: TDateTime): Longint;
4759: { The same as previous but if Date2 < Dat1 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;
const ToCode, FromCode: AnsiString): Boolean;
4782: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4783: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4784: function OemStrToAnsi(const S: AnsiString): AnsiString;
4785: function AnsiStrToOem(const S: AnsiString): AnsiString;
4786: {$ENDIF UNIX}
4787: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4788: { StrToOem translates a string from the Windows character set into the OEM character set. }
4789: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4790: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4791: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4792:

```

```

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 beginning to first character Symb. }
4836: function Copy2SymbDel(var S: string; Symb: Char): string;
4837: { Copy2SymbDel returns a substring of a string S from beginning 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 beginning to first white space. }
4841: function Copy2SpaceDel(var S: string): string;
4842: { Copy2SpaceDel returns a substring of a string S from beginning 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: {$ENDIF MSWINDOWS}
5026: procedure ResourceNotFound(ResID: PChar);
5027: function EmptyRect: TRect;
5028: function RectWidth(R: TRect): Integer;
5029: function RectHeight(R: TRect): Integer;
5030: function CompareRect(const R1, R2: TRect): Boolean;
5031: procedure RectNormalize(var R: TRect);
5032: function RectIsSquare(const R: TRect): Boolean;
5033: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5034: //If AMaxSize = -1 ,then auto calc Square's max size
5035: {$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( hWindow : 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: 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: 'ESBLnPi', '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 IGratestPowerOf2( const N : LongWord) : LongWord
5392: Function ESBArcCosh( X : Extended) : Extended
5393: Function ESBArcSinh( X : Extended) : Extended
5394: Function ESBArcTanh( 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:int;Pool1:TMemoryStreamPo
Numbers
5425: Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5426:
5427: procedure SIRegister_xrtrl_math_Integer(CL: TPSPascalCompiler);
5428: begin
5429:   AddTypeS('TXRTLInteger', 'array of Integer
5430:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5431:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5432:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5433:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5434:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5435:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5436:   AddClassN(FindClass('TOBJECT'),'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,
AHighApproxResult:TXRTLInteger)
5478: Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHighApproxResult: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 XRTLRCDDR(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 JvXPMMethodsEqual( 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 ACanvas: 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_utexplore(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:ShortString;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:SString;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 ByteFlagsIsSet( 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:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5724: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,
Redemption:Ext;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:Int;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:Int;Pmt,PV,
FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
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 DevSql6( 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( Datel, Date2 : TStDate; var Days, Months, Years : Integer)
5879: Function BondDateDiff( Datel, 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 DateTimeHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5909: Function MonthToString( const Month : Integer) : string
5910: Function DateTimeToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5911: Function DateTimeToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5912: Function StDateToDateTime( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5913: Function DayOfWeekToString( const WeekDay : TStDayType) : string
5914: Function DMYtoDateTime(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5915: Function CurrentDateTime( 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 DateTimeIsBlank( 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 HugeCompressRLE( 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 LoCase( 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_usvd(CL: TPSPascalCompiler);
5957: begin
5958: Procedure SV_DeComp( 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;Neqn: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:Ptr;AParam:Pointer;ASafeSection:bool;
6084: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6085: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool;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:Card;const ByteBufferSpace:Card):
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; UriIs : TUriIs) : 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_MSIDESCRIPTOR', '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 : UINT) : 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 RRegister_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; pszGroupName
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
FunctionName: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):Bool
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):Bool
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):Bool;
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'),'EHTTPParser
6644: //AnsiCharSet', 'set of AnsiChar
6645: AnsiStringArray', 'array of AnsiString
6646: THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6647: THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
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: + 'anguage, 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: //PHTPCustomHeader', '^THTPCustomHeader // will not work

```

```

6665: THHTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6666: THHTTPContentLength', 'record Value : THHTTPContentLengthEnum; ByteCount: Int64; end
6667: //PHTTTPContentLength', '^THHTTPContentLength // will not work
6668: THHTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6669: THHTTPContentTypeEnum', '( 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: THHTTPContentType', 'record Value : THHTTPContentTypeEnum; CustomM
6675:   + 'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;
6676:   + ' CustomStr : AnsiString; end
6677: THHTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6678: THHTTPDateField', 'record Value : THHTTPDateFieldEnum; 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: THHTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6683: THHTTPTransferEncoding', 'record Value : THHTTPTransferEncodingEnu
6684:   + 'm; Custom : AnsiString; end
6685: THHTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6686: THHTTPConnectionField', 'record Value : THHTTPConnectionFieldEnum;
6687:   + ' Custom : AnsiString; end
6688: THHTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6689: THHTTPAgeField', 'record Value : THHTTPAgeFieldEnum; Age : Int64; Custom: AnsiString; end
6690: THHTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6691: THHTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore
6692:   + ', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6693: THHTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate
6694:   + ', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf
6695:   + ' ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6696: THHTTPCacheControlField', 'record Value : THHTTPCacheControlFieldEnum; end
6697: THHTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h
6698:   + 'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6699: THHTTPContentEncoding', 'record Value: THHTTPContentEncodingEnum; Custom: AnsiString; end;
6700: THHTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6701: THHTTPContentEncodingField', 'record Value : THHTTPContentEncoding
6702:   + 'FieldEnum; List : array of THHTTPContentEncoding; end
6703: THHTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6704: THHTTPRetryAfterField', 'record Value : THHTTPRetryAfterFieldEnum;
6705:   + ' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6706: THHTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6707: THHTTPContentRangeField', 'record Value : THHTTPContentRangeFieldE
6708:   + 'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6709: THHTTPSetCookieFieldEnum', '( hskoNone, hskoDecoded, hskoCustom )
6710: THHTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6711: THHTTPSetCookieCustomFieldArray', 'array of THHTTPSetCookieCustomField
6712: THHTTPSetCookieField', 'record Value : THHTTPSetCookieFieldEnum; D
6713:   + 'omain : AnsiString; Path : AnsiString; Expires : THHTTPDateField; MaxAge :
6714:   + 'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THHTTPSetCookie
6715:   + 'CustomFieldArray; Custom : AnsiString; end
6716: //PHTTTPSetCookieField', '^THHTTPSetCookieField // will not work
6717: THHTTPSetCookieFieldArray', 'array of THHTTPSetCookieField
6718: THHTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6719: THHTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6720: //PHTTTPCookieFieldEntry', '^THHTTPCookieFieldEntry // will not work
6721: THHTTPCookieFieldEntryArray', 'array of THHTTPCookieFieldEntry
6722: THHTTPCookieField', 'record Value : THHTTPCookieFieldEnum; Entries
6723:   + ' : THHTTPCookieFieldEntryArray; Custom : AnsiString; end
6724: THHTTPCommonHeaders', 'record TransferEncoding : THHTTPTransferEnc
6725:   + 'oding; ContentType : THHTTPContentType; ContentLength : THHTTPContentLength;
6726:   + ' Connection : THHTTPConnectionField; ProxyConnection : THHTTPConnectionField
6727:   + ' Date : THHTTPDateField; ContentEncoding : THHTTPContentEncodingField; end
6728: THHTTPCustomHeaders', 'array of THHTTPCustomHeader
6729:   //THHTTPFixedHeaders', 'array[THHTTPHeaderNameEnum] of AnsiString
6730: THHTTPFixedHeaders', 'array[0..42] of AnsiString
6731:   THHTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC
6732:   + 'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6733: THHTTPMethod', 'record Value : THHTTPMethodEnum; Custom : AnsiString; end
6734: THHTTPRequestStartLine', 'record Method: THHTTPMethod; URI: AnsiString; Version: THHTTPVersion; end
6735: THHTTPRequestHeader', 'record CommonHeaders : THHTTPCommonHeaders;
6736:   + ' FixedHeaders : THHTTPFixedHeaders; CustomHeaders : THHTTPCustomHeaders; Co
6737:   + 'okie : THHTTPCookieField; IfModifiedSince: THHTTPDateField; IfUnmodifiedSince: THHTTPDateField; end
6738: //PHTTTPRequestHeader', '^THHTTPRequestHeader // will not work
6739: THHTTPRequest', 'record StartLine : THHTTPRequestStartLine; Header
6740:   + ' : THHTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6741: THHTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK )
6742: THHTTPResponseStartLine', 'record Version : THHTTPVersion; Code :
6743:   + ' Integer; Msg : THHTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6744: THHTTPResponseHeader', 'record CommonHeaders : THHTTPCommonHeaders
6745:   + ' FixedHeaders : THHTTPFixedHeaders; CustomHeaders : THHTTPCustomHeaders; Co
6746:   + 'okies : THHTTPSetCookieFieldArray; Expires : THHTTPDateField; LastModified :
6747:   + ' THHTTPDateField; Age : THHTTPAgeField; end
6748: //PHTTTPResponseHeader', '^THHTTPResponseHeader // will not work
6749: THHTTPResponse', 'record StartLine : THHTTPResponseStartLine; Head
6750:   + 'er : THHTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6751: Function HTTPMessageHasContent( const H : THHTTPCommonHeaders ) : Boolean
6752: Procedure InitHTTPRequest( var A : THHTTPRequest )
6753: Procedure InitHTTPResponse( var A : THHTTPResponse )

```

```

6754: Procedure ClearHTTPVersion( var A : THTTPVersion)
6755: Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6756: Procedure ClearHTTPContentType( var A : THTTPContentType)
6757: Procedure ClearHTTPDateField( var A : THTTPDateField)
6758: Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6759: Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6760: Procedure ClearHTTPPageField( var A : THTTPPageField)
6761: Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6762: Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6763: Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6764: Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6765: Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6766: //Procedure ClearHTTFFixedHeaders( var A : THTTFFixedHeaders)
6767: Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6768: Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6769: Procedure ClearHTTPMethod( var A : THTTPMethod)
6770: Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6771: Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6772: Procedure ClearHTTPRequest( var A : THTTPRequest)
6773: Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6774: Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6775: Procedure ClearHTTPResponse( var A : THTTPResponse)
6776:   THTTPStringOption', '( hsoNone )
6777:   THTTPStringOptions', 'set of THTTPStringOption
6778:   FindClass('TOBJECT'),'TAnsiStringBuilder
6779:
6780: Procedure BuildStrHTTPVersion(const A:THTTPVersion; const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6781: Procedure BuildStrHTTPContentLengthValue( const A : THTTPContentLength; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6782: Procedure BuildStrHTTPContentLength( const A : THTTPContentLength; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6783: Procedure BuildStrHTTPContentTypeValue( const A : THTTPContentType; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6784: Procedure BuildStrHTTPContentType(const A:THTTPContentType;const B:TAnsiStringBuilder; const
P:THTTPStringOptions)
6785: Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
B : TAnsiStringBuilder; const P : THTTPStringOptions)
6786: Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6787: Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6788: Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
TAnsiStringBuilder; const P : THTTPStringOptions)
6789: Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6790: Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6791: Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6792: Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6793: Procedure BuildStrHTTPPageField(const A:THTTPPageField;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6794: Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6795: Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
B:TAnsiStringBuilder;const P:THTTPStringOptions)
6796: Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6797: Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6798: Procedure BuildStrHTTFFixedHeaders(const A:THTTFFixedHeaders;const B:TAnsiStrBuilder;const
P:THTTPStringOptions)
6799: Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6800: Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6801: Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6802: Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6803: Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6804: Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6805: Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6806: Procedure BuildStrHTTPRequest( const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6807: Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
TAnsiStringBuilder; const P : THTTPStringOptions)
6808: Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
THTTPStrOptions);
6809: Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
P:THTTPStringOptions);
6810: Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
P:THTTPStringOptions);
6811: Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString

```

```

6812: Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6813: Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6814: Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6815: Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6816: Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6817: Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
Domain:AnsiString;const Secure:Bool;
6818:   THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT
6819:   +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6820:   SIRegister_THTTPParser(CL);
6821:   FindClass('TOBJECT'),'THTTPContentDecoder
6822:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6823:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsize )
6824:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
6825:   + ' 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: procedure SIRegister_cTlsUtils(CL: TPSPascalCompiler);
6842: begin
6843:   'TLSSLibraryVersion', 'String').SetString( '1.00
6844:   'TLSError_None', 'LongInt').SetInt( 0);
6845:   'TLSError_InvalidBuffer', 'LongInt').SetInt( 1);
6846:   'TLSError_InvalidParameter', 'LongInt').SetInt( 2);
6847:   'TLSError_InvalidCertificate', 'LongInt').SetInt( 3);
6848:   'TLSError_InvalidState', 'LongInt').SetInt( 4);
6849:   'TLSError_DecodeError', 'LongInt').SetInt( 5);
6850:   'TLSError_BadProtocol', 'LongInt').SetInt( 6);
6851:   Function TLSErrorMessage( const TLSError : Integer) : String
6852:   SIRegister_ETLSError(CL);
6853:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6854:   TTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6855:   Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6856:   Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6857:   Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6858:   Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6859:   Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6860:   Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6861:   Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6862:   Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6863:   Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6864:   Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6865:   Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6866:   Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6867:   Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6868:   Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6869:   Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6870:   Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6871:   Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6872:   PTLSRandom', '^TTLSRandom // will not work
6873:   Procedure InitTLSRandom( var Random : TTLSRandom)
6874:   Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6875:   'TLSSessionIDMaxLen', 'LongInt').SetInt( 32);
6876:   Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6877:   Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6878:   Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6879:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6880:   + ' Signature : TTLSSignatureAlgorithm; end
6881:   // TTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +// will not work
6882:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6883:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_
6884:   + 'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6885:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma
6886:   + 'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6887:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I
6888:   + 'integer; Supported : Boolean; end
6889:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6890:   'TLS_MAC_MAXDIGESTSIZE', 'LongInt').SetInt( 64);
6891:   TTLSPRFAlgorithm', '( tlpshaSHA256 )
6892:   Function tlp_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6893:   Function tlp_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6894:   Function tlp_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6895:   Function tlp_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6896:   Function tlp10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6897:   Function tlp12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6898:   Function tlp12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString

```



```

6899: Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
Size:Int):AString;
6900: Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:Ansistring; const
Size:Integer):Ansistring;
6901: Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: Ansistring;const
Size:Int):Ansistring;
6902: Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: Ansistring;const
Size:Int):Ansistring;
6903: Function TLSKeyBlock( const ProtocolVersion : TTLSProtocolVersion; const MasterSecret, ServerRandom,
ClientRandom : Ansistring; const Size : Integer) : Ansistring;
6904: Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:Ansistring) : Ansistring;
6905: Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:Ansistring):Ansistring;
6906: Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:Ansistring): Ansistring;
6907: Function TLSPMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
ServerRandom:Ansistring) : Ansistring;
6908: Function TLSKeys, record KeyBlock : Ansistring; ClientMACKey : Ansistr
6909: ing; ServerMACKey : Ansistring; ClientEncKey : Ansistring; ServerEncKey : '
6910: ing; ClientIV : Ansistring; ServerIV : Ansistring; end
6911: Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : Ansistring; var TLSKeys : TLSKeys)
6912: Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:Ansistring;var TLSKeys:TLSKeys)
6913: 'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt').SetInt( 16384 - 1);
6914: 'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt').SetInt( 16384 + 1024);
6915: Procedure SelfTestcTLSUtils
6916: end;
6917:
6918: (*-----*)
6919: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6920: begin
6921: sPosData,record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
integer; disks : integer; mx : integer; my : integer; end
6922: // pBoard', '^tBoard // will not work
6923: Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6924: Function rCheckMove( color : byte; cx, cy : integer) : integer
6925: //Function rDoStep( data : pBoard) : word
6926: Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6927: end;
6928:
6929: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6930: begin
6931: Function InEditMode( ADataset : TDataset) : Boolean
6932: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6933: Function CheckDataSourceel(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6934: Function GetFieldText( AField : TField) : String
6935: end;
6936:
6937: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6938: begin
6939: TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6940: TMyPrintRange', '( prAll, prSelected )
6941: TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6942: + 'ded, ssDateTime, ssTime, ssCustom )
6943: TSortDirection', '( sdAscending, sdDescending )
6944: TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6945: TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
6946: + 'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6947: TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6948: TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
6949: SIRegister_TSortOptions(CL);
6950: SIRegister_TPrintOptions(CL);
6951: TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6952: SIRegister_TSortedList(CL);
6953: TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6954: TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6955: TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6956: TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
6957: + 'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
6958: SIRegister_TFontSetting(CL);
6959: SIRegister_TFontList(CL);
6960: AddTypesS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row : '
6961: + 'integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
6962: TSetFilterEvent', 'Procedure ( ARows : TStringList; var Accept : Boolean)
6963: TSearchEvent', 'Procedure ( ARows : TStringList; var Accept : Boolean)
6964: TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
6965: TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
6966: TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
6967: TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
6968: TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : integer'
6969: + 'r; var SortStyle : TSortStyle)
6970: TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow : '
6971: + 'integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
6972: SIRegister_TSortGrid(CL);
6973: Function ExtendedCompare( const Str1, Str2 : String) : Integer
6974: Function NormalCompare( const Str1, Str2 : String) : Integer
6975: Function DateTimeCompare( const Str1, Str2 : String) : Integer
6976: Function NumericCompare( const Str1, Str2 : String) : Integer
6977: Function TimeCompare( const Str1, Str2 : String) : Integer
6978: //Function Compare( Item1, Item2 : Pointer) : Integer

```

```

6979: end;
6980:
6981: ***** procedure Register_IB(CL: TPSPascalCompiler);
6982: Procedure IBAAlloc( var P, OldSize, NewSize : Integer)
6983: Procedure IBERror( ErrMess : TIBClientError; const Args : array of const)
6984: Procedure IBDataBaseError
6985: Function StatusVector : PISC_STATUS
6986: Function StatusVectorArray : PStatusVector
6987: Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
6988: Function StatusVectorAsText : string
6989: Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
6990: Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
6991:
6992:
6993: //*****unit uPSI_BoldUtils;*****
6994: Function CharCount( c : char; const s : string) : integer
6995: Function BoldNamesEqual( const name1, name2 : string) : Boolean
6996: Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
6997: Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
6998: Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
6999: Function BoldTrim( const S : string) : string
7000: Function BoldIsPrefix( const S, Prefix : string) : Boolean
7001: Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7002: Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7003: Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7004: Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7005: Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7006: //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7007: Function CapitalisedToSpaced( Capitalised : string) : string
7008: Function SpacedToCapitalised( Spaced : string) : string
7009: Function BooleanToString( BoolValue : Boolean) : string
7010: Function StringToBoolean( StrValue : string) : Boolean
7011: Function GetUpperLimitForMultiplicity( const Multiplicity : string) : Integer
7012: Function GetLowerLimitForMultiplicity( const Multiplicity : string) : Integer
7013: Function StringListToVarArray( List : TStringList) : variant
7014: Function IsLocalMachine( const MachineName : WideString) : Boolean
7015: Function GetComputerNameStr : string
7016: Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7017: Function BoldStrToDateFmt( const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7018: Function BoldDateToStrFmt( const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):string;
7019: Function BoldParseFormattedDateList( const value:string;const formats:TStrings;var Date:TDateTime):Boolean;
7020: Function BoldParseFormattedDate( const value:string;const formats:array of string; var
Date:TDateTime):Boolean;
7021: Procedure EnsureTrailing( var Str : string; ch : char)
7022: Function BoldDirectoryExists( const Name : string) : Boolean
7023: Function BoldForceDirectories( Dir : string) : Boolean
7024: Function BoldRootRegistryKey : string
7025: Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7026: Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7027: Function LogicalAnd( A, B : Integer) : Boolean
7028: record TByHandleFileInformation dwFileAttributes : DWORD; '
7029: +ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7030: +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7031: +eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7032: Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7033: Function IsFirstInstance : Boolean
7034: Procedure ActivateFirst( AString : PChar)
7035: Procedure ActivateFirstCommandLine
7036: function MakeAckPkt(const BlockNumber: Word): string;
7037: procedure SendError(UDPBase:TiUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7038: procedure SendError(UDPClient: TiUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7039: procedure SendError(UDPBase: TiUDPBase; APeerIP: string; const APort: Integer; E: Exception); overload;
7040: procedure SendError(UDPClient: TiUDPClient; E: Exception); overload;
7041: function IdStrToWord(const Value: string): Word;
7042: function IdWordToStr(const Value: Word): WordStr;
7043: Function HasInstructionSet( const InstructionSet : TCPUIInstructionSet) : Boolean
7044: Function CPUFeatures : TCPUFeatures
7045:
7046:
7047: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7048: begin
7049: AddTypes('TXRTLBitIndex', 'Integer
7050: Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7051: Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7052: Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7053: Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7054: Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7055: Function XRTLSwapHiLo16( X : Word) : Word
7056: Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7057: Function XRTLSwapHiLo64( X : Int64) : Int64
7058: Function XRTLROL32( A, S : Cardinal) : Cardinal
7059: Function XRTLROL32( A, S : Cardinal) : Cardinal
7060: Function XRTLROL16( A : Word; S : Cardinal) : Word
7061: Function XRTLROL16( A : Word; S : Cardinal) : Word
7062: Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7063: Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7064: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7065: //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7066: Procedure XRTLUMul16( const A, B : Integer; var MulL, MulH : Integer)

```

```

7067: Function XRTLPopulation( A : Cardinal ) : Cardinal
7068: end;
7069:
7070: Function XRTLURLDecode( const ASrc : WideString ) : WideString
7071: Function XRTLURLEncode( const ASrc : WideString ) : WideString
7072: Function XRTLURINormalize( const AURI : WideString ) : WideString
7073: Procedure XRTLURIParse(const AURI:Widestring;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
VPassword : WideString)
7074: Function XRTLExtractLongPathName(APath: string): string;
7075:
7076: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7077: begin
7078:   AddTypeS('Int8', 'ShortInt
7079:   AddTypeS('Int16', 'SmallInt
7080:   AddTypeS('Int32', 'LongInt
7081:   AddTypeS('UInt8', 'Byte
7082:   AddTypeS('UInt16', 'Word
7083:   AddTypeS('UInt32', 'LongWord
7084:   AddTypeS('UInt64', 'Int64
7085:   AddTypeS('Word8', 'UInt8
7086:   AddTypeS('Word16', 'UInt16
7087:   AddTypeS('Word32', 'UInt32
7088:   AddTypeS('Word64', 'UInt64
7089:   AddTypeS('LargeInt', 'Int64
7090:   AddTypeS('NativeInt', 'Integer
7091:   AddTypeS('NativeUInt', 'Cardinal
7092:   Const('BitsPerByte','LongInt').SetInt( 8);
7093:   Const('BitsPerWord','LongInt').SetInt( 16);
7094:   Const('BitsPerLongWord','LongInt').SetInt( 32);
7095:   //Const('BitsPerCardinal','LongInt').SetInt( BytesPerCardinal * 8);
7096:   //Const('BitsPerNativeWord','LongInt').SetInt( BytesPerNativeWord * 8);
7097:   Function MinI( const A, B : Integer ) : Integer
7098:   Function MaxI( const A, B : Integer ) : Integer
7099:   Function MinC( const A, B : Cardinal ) : Cardinal
7100:   Function MaxC( const A, B : Cardinal ) : Cardinal
7101:   Function SumClipI( const A, I : Integer ) : Integer
7102:   Function SumClipC( const A : Cardinal; const I : Integer ) : Cardinal
7103:   Function InByteRange( const A : Int64 ) : Boolean
7104:   Function InWordRange( const A : Int64 ) : Boolean
7105:   Function InLongWordRange( const A : Int64 ) : Boolean
7106:   Function InShortIntRange( const A : Int64 ) : Boolean
7107:   Function InSmallIntRange( const A : Int64 ) : Boolean
7108:   Function InLongIntRange( const A : Int64 ) : Boolean
7109:   AddTypeS('Bool8', 'ByteBool
7110:   AddTypeS('Bool16', 'WordBool
7111:   AddTypeS('Bool32', 'LongBool
7112:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7113:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7114:   Function ReverseCompareResult( const C : TCompareResult ) : TCompareResult
7115:   Const('MinSingle','Single').setExtended( 1.5E-45);
7116:   Const('MaxSingle','Single').setExtended( 3.4E+38);
7117:   Const('MinDouble','Double').setExtended( 5.0E-324);
7118:   Const('MaxDouble','Double').setExtended( 1.7E+308);
7119:   Const('MinExtended','Extended').setExtended(3.4E-4932);
7120:   Const('MaxExtended','Extended').setExtended(1.1E+4932);
7121:   Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7122:   Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7123:   Function MinF( const A, B : Float ) : Float
7124:   Function MaxF( const A, B : Float ) : Float
7125:   Function ClipF( const Value : Float; const Low, High : Float ) : Float
7126:   Function InSingleRange( const A : Float ) : Boolean
7127:   Function InDoubleRange( const A : Float ) : Boolean
7128:   Function InCurrencyRange( const A : Float ) : Boolean;
7129:   Function InCurrencyRangel( const A : Int64 ) : Boolean;
7130:   Function FloatExponentBase2( const A : Extended; var Exponent : Integer ) : Boolean
7131:   Function FloatExponentBase10( const A : Extended; var Exponent : Integer ) : Boolean
7132:   Function FloatIsInfinity( const A : Extended ) : Boolean
7133:   Function FloatIsNaN( const A : Extended ) : Boolean
7134:   Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7135:   Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7136:   Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7137:   Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7138:   Function FloatZero( const A : Float; const CompareDelta : Float ) : Boolean
7139:   Function FloatOne( const A : Float; const CompareDelta : Float ) : Boolean
7140:   Function FloatsEqual( const A, B : Float; const CompareDelta : Float ) : Boolean
7141:   Function FloatsCompare( const A, B : Float; const CompareDelta : Float ) : TCompareResult
7142:   Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7143:   Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7144:   Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7145:   Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7146:   Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double ) : Boolean
7147:   Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double ) : TCompareResult
7148:   Function cClearBit( const Value, BitIndex : LongWord ) : LongWord
7149:   Function cSetBit( const Value, BitIndex : LongWord ) : LongWord
7150:   Function cIsBitSet( const Value, BitIndex : LongWord ) : Boolean
7151:   Function cToggleBit( const Value, BitIndex : LongWord ) : LongWord
7152:   Function cIsHighBitSet( const Value : LongWord ) : Boolean
7153:   Function SetBitScanForward( const Value : LongWord ) : Integer;
7154:   Function SetBitScanForward1( const Value, BitIndex : LongWord ) : Integer;

```

```

7155: Function SetBitScanReverse( const Value : LongWord) : Integer;
7156: Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7157: Function ClearBitScanForward( const Value : LongWord) : Integer;
7158: Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7159: Function ClearBitScanReverse( const Value : LongWord) : Integer;
7160: Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7161: Function cReverseBits( const Value : LongWord) : LongWord;
7162: Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7163: Function cSwapEndian( const Value : LongWord) : LongWord;
7164: Function cTwosComplement( const Value : LongWord) : LongWord;
7165: Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word;
7166: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord;
7167: Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word;
7168: Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord;
7169: Function cBitCount( const Value : LongWord) : LongWord;
7170: Function cIsPowerOfTwo( const Value : LongWord) : Boolean;
7171: Function LowBitMask( const HighBitIndex : LongWord) : LongWord;
7172: Function HighBitMask( const LowBitIndex : LongWord) : LongWord;
7173: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord;
7174: Function SetBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord;
7175: Function ClearBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord;
7176: Function ToggleBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord;
7177: Function IsBitRangeSet( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean;
7178: Function IsBitRangeClear( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean;
7179: // AddTypes('CharSet', 'set of AnsiChar
7180: AddTypes('CharSet', 'set of Char' //!!!
7181: AddTypes('AnsiCharSet', 'TCharSet
7182: AddTypes('ByteSet', 'set of Byte
7183: AddTypes('AnsiChar', 'Char
7184: // Function AsCharSet( const C : array of AnsiChar) : CharSet
7185: Function AsByteSet( const C : array of Byte) : ByteSet;
7186: Procedure ComplementChar( var C : CharSet; const Ch : Char);
7187: Procedure ClearCharSet( var C : CharSet);
7188: Procedure FillCharSet( var C : CharSet);
7189: Procedure ComplementCharSet( var C : CharSet);
7190: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet);
7191: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet);
7192: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet);
7193: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet);
7194: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet);
7195: Function IsSubSet( const A, B : CharSet) : Boolean;
7196: Function IsEqual( const A, B : CharSet) : Boolean;
7197: Function IsEmpty( const C : CharSet) : Boolean;
7198: Function IsComplete( const C : CharSet) : Boolean;
7199: Function cCharCount( const C : CharSet) : Integer;
7200: Procedure ConvertCaseInsensitive( var C : CharSet);
7201: Function CaseInsensitiveCharSet( const C : CharSet) : CharSet;
7202: Function IntRangeLength( const Low, High : Integer) : Int64;
7203: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean;
7204: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean;
7205: Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean;
7206: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean;
7207: Function IntRangeIncludeElementRange( var Low, High : Integer; const LowElement, HighElement : Integer) : Boolean;
7208: Function CardRangeLength( const Low, High : Cardinal) : Int64;
7209: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean;
7210: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean;
7211: Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean;
7212: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean;
7213: Function CardRangeIncludeElementRange( var Low, High : Cardinal; const LowElement, HighElement : Cardinal) : Boolean;
7214: AddTypes('UnicodeChar', 'WideChar
7215: Function Compare( const I1, I2 : Boolean) : TCompareResult;
7216: Function Compare1( const I1, I2 : Integer) : TCompareResult;
7217: Function Compare2( const I1, I2 : Int64) : TCompareResult;
7218: Function Compare3( const I1, I2 : Extended) : TCompareResult;
7219: Function CompareA( const I1, I2 : AnsiString) : TCompareResult;
7220: Function CompareW( const I1, I2 : WideString) : TCompareResult;
7221: Function cSgn( const A : LongInt) : Integer;
7222: Function cSgn1( const A : Int64) : Integer;
7223: Function cSgn2( const A : Extended) : Integer;
7224: AddTypes('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7225: Function AnsiCharToInt( const A : AnsiChar) : Integer;
7226: Function WideCharToInt( const A : WideChar) : Integer;
7227: Function CharToInt( const A : Char) : Integer;
7228: Function IntToAnsiChar( const A : Integer) : AnsiChar;
7229: Function IntToWideChar( const A : Integer) : WideChar;
7230: Function IntToChar( const A : Integer) : Char;
7231: Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean;
7232: Function IsHexWideChar( const Ch : WideChar) : Boolean;
7233: Function IsHexChar( const Ch : Char) : Boolean;
7234: Function HexAnsiCharToInt( const A : AnsiChar) : Integer;
7235: Function HexWideCharToInt( const A : WideChar) : Integer;
7236: Function HexCharToInt( const A : Char) : Integer;
7237: Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar;
7238: Function IntToUpperHexWideChar( const A : Integer) : WideChar;
7239: Function IntToUpperHexChar( const A : Integer) : Char;
7240: Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar;
7241: Function IntToLowerHexWideChar( const A : Integer) : WideChar;
7242: Function IntToLowerHexChar( const A : Integer) : Char;
7243: Function IntToStringA( const A : Int64) : AnsiString;

```



```

7244: Function IntToStringW( const A : Int64 ) : WideString
7245: Function IntToString( const A : Int64 ) : String
7246: Function UIntToStringA( const A : NativeUInt ) : AnsiString
7247: Function UIntToStringW( const A : NativeUInt ) : WideString
7248: Function UIntToString( const A : NativeUInt ) : String
7249: Function LongWordToStrA( const A : LongWord; const Digits : Integer ) : AnsiString
7250: Function LongWordToStrW( const A : LongWord; const Digits : Integer ) : WideString
7251: Function LongWordToStrU( const A : LongWord; const Digits : Integer ) : UnicodeString
7252: Function LongWordToStr( const A : LongWord; const Digits : Integer ) : String
7253: Function LongWordToHexA( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7254: Function LongWordToHexW( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7255: Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7256: Function LongWordToOctA( const A : LongWord; const Digits : Integer ) : AnsiString
7257: Function LongWordToOctW( const A : LongWord; const Digits : Integer ) : WideString
7258: Function LongWordToOct( const A : LongWord; const Digits : Integer ) : String
7259: Function LongWordToBinA( const A : LongWord; const Digits : Integer ) : AnsiString
7260: Function LongWordToBinW( const A : LongWord; const Digits : Integer ) : WideString
7261: Function LongWordToBin( const A : LongWord; const Digits : Integer ) : String
7262: Function TryStringToInt64A( const S : AnsiString; out A : Int64 ) : Boolean
7263: Function TryStringToInt64W( const S : WideString; out A : Int64 ) : Boolean
7264: Function TryStringToInt64( const S : String; out A : Int64 ) : Boolean
7265: Function StringToInt64DefA( const S : AnsiString; const Default : Int64 ) : Int64
7266: Function StringToInt64DefW( const S : WideString; const Default : Int64 ) : Int64
7267: Function StringToInt64Def( const S : String; const Default : Int64 ) : Int64
7268: Function StringToInt64A( const S : AnsiString ) : Int64
7269: Function StringToInt64W( const S : WideString ) : Int64
7270: Function StringToInt64( const S : String ) : Int64
7271: Function TryStringToIntA( const S : AnsiString; out A : Integer ) : Boolean
7272: Function TryStringToIntW( const S : WideString; out A : Integer ) : Boolean
7273: Function TryStringToInt( const S : String; out A : Integer ) : Boolean
7274: Function StringToIntDefA( const S : AnsiString; const Default : Integer ) : Integer
7275: Function StringToIntDefW( const S : WideString; const Default : Integer ) : Integer
7276: Function StringToIntDef( const S : String; const Default : Integer ) : Integer
7277: Function StringToIntA( const S : AnsiString ) : Integer
7278: Function StringToIntW( const S : WideString ) : Integer
7279: Function StringToInt( const S : String ) : Integer
7280: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7281: Function TryStringToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7282: Function TryStringToLongWord( const S : String; out A : LongWord ) : Boolean
7283: Function StringToLongWordA( const S : AnsiString ) : LongWord
7284: Function StringToLongWordW( const S : WideString ) : LongWord
7285: Function StringToLongWord( const S : String ) : LongWord
7286: Function HexToUIntA( const S : AnsiString ) : NativeUInt
7287: Function HexToUIntW( const S : WideString ) : NativeUInt
7288: Function HexToUInt( const S : String ) : NativeUInt
7289: Function TryHexToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7290: Function TryHexToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7291: Function TryHexToLongWord( const S : String; out A : LongWord ) : Boolean
7292: Function HexToLongWordA( const S : AnsiString ) : LongWord
7293: Function HexToLongWordW( const S : WideString ) : LongWord
7294: Function HexToLongWord( const S : String ) : LongWord
7295: Function TryOctToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7296: Function TryOctToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7297: Function TryOctToLongWord( const S : String; out A : LongWord ) : Boolean
7298: Function OctToLongWordA( const S : AnsiString ) : LongWord
7299: Function OctToLongWordW( const S : WideString ) : LongWord
7300: Function OctToLongWord( const S : String ) : LongWord
7301: Function TryBinToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7302: Function TryBinToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7303: Function TryBinToLongWord( const S : String; out A : LongWord ) : Boolean
7304: Function BinToLongWordA( const S : AnsiString ) : LongWord
7305: Function BinToLongWordW( const S : WideString ) : LongWord
7306: Function BinToLongWord( const S : String ) : LongWord
7307: Function FloatToStringA( const A : Extended ) : AnsiString
7308: Function FloatToStringW( const A : Extended ) : WideString
7309: Function FloatToString( const A : Extended ) : String
7310: Function TryStringToFloatA( const A : AnsiString; out B : Extended ) : Boolean
7311: Function TryStringToFloatW( const A : WideString; out B : Extended ) : Boolean
7312: Function TryStringToFloat( const A : String; out B : Extended ) : Boolean
7313: Function StringToFloatA( const A : AnsiString ) : Extended
7314: Function StringToFloatW( const A : WideString ) : Extended
7315: Function StringToFloat( const A : String ) : Extended
7316: Function StringToFloatDefA( const A : AnsiString; const Default : Extended ) : Extended
7317: Function StringToFloatDefW( const A : WideString; const Default : Extended ) : Extended
7318: Function StringToFloatDef( const A : String; const Default : Extended ) : Extended
7319: Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean; const PadMultiple:Integer; const
PadChar: AnsiChar ) : AnsiString
7320: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet ) : AnsiString
7321: unit uPSI_cFundamentUtils;
7322: Const ('b64_MIMEBase64', 'Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7323: Const ('b64_UUEncode', 'String').String('!\"#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMN
7324: Const ('b64_XXEncode', 'String').String('+-0123456789ABCDEFGHIJKLMN
7325: Const ('CCHARSET', 'String').SetString(b64_XXEncode);
7326: Const ('CHEXSET', 'String').SetString('0123456789ABCDEF
7327: Const ('HEXDIGITS', 'String').SetString('0123456789ABCDEF
7328: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7329: Const ('DIGISET', 'String').SetString('0123456789
7330: Const ('LETTERSET', 'String').SetString('ABCDEFGHIJKLMN
7331: Const ('DIGISET2', 'TCharSet').SetSet('0123456789'

```

```

7332: Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ')
7333: Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7334: Const('NUMBERSET','TCharset').SetSet('0123456789');
7335: Const('NUMBERS','String').SetString('0123456789');
7336: Const('LETTERS','String').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7337: Function CharSetToStr( const C : CharSet ) : AnsiString
7338: Function StrToCharSet( const S : AnsiString ) : CharSet
7339: Function MIMEBase64Decode( const S : AnsiString ) : AnsiString
7340: Function MIMEBase64Encode( const S : AnsiString ) : AnsiString
7341: Function UUDecode( const S : AnsiString ) : AnsiString
7342: Function XXDecode( const S : AnsiString ) : AnsiString
7343: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean ) : AnsiString
7344: Function InterfaceToStrA( const I : IInterface ) : AnsiString
7345: Function InterfaceToStrW( const I : IInterface ) : WideString
7346: Function InterfaceToStr( const I : IInterface ) : String
7347: Function ObjectClassName( const O : TObject ) : String
7348: Function ClassClassName( const C : TClass ) : String
7349: Function ObjectToStr( const O : TObject ) : String
7350: Function ObjectToString( const O : TObject ) : String
7351: Function CharSetToStr( const C : CharSet ) : AnsiString
7352: Function StrToCharSet( const S : AnsiString ) : CharSet
7353: Function HashStrA( const S : AnsiString; const Index : Integer; const Count : Integer; const
  AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7354: Function HashStrW( const S : WideString; const Index : Integer; const Count : Integer; const
  AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7355: Function HashStr( const S : String; const Index : Integer; const Count : Integer; const
  AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7356: Function HashInteger( const I : Integer; const Slots : LongWord ) : LongWord
7357: Function HashLongWord( const I : LongWord; const Slots : LongWord ) : LongWord
7358: Const('Bytes1KB','LongInt').SetInt( 1024);
7359: SIRegister_IInterface(CL);
7360: Procedure SelfTestCFundamentUtils
7361:
7362: Function CreateSchedule : IJclSchedule
7363: Function NullStamp : TTimeStamp
7364: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
7365: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
7366: Function IsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean
7367:
7368: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7369: begin
7370:   AddTypeS('TFunc', 'function(X : Float) : Float;
7371: Function InitGraphics( Width, Height : Integer ) : Boolean
7372: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7373: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7374: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7375: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7376: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7377: Procedure SetGraphTitle( Title : String)
7378: Procedure SetOxTitle( Title : String)
7379: Procedure SetOyTitle( Title : String)
7380: Function GetGraphTitle : String
7381: Function GetOxTitle : String
7382: Function GetOyTitle : String
7383: Procedure PlotOxAxis( Canvas : TCanvas)
7384: Procedure PlotOyAxis( Canvas : TCanvas)
7385: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7386: Procedure WriteGraphTitle( Canvas : TCanvas)
7387: Function SetMaxCurv( NCurv : Byte ) : Boolean
7388: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7389: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7390: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7391: Procedure SetCurvStep( CurvIndex, Step : Integer)
7392: Function GetMaxCurv : Byte
7393: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7394: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7395: Function GetCurvLegend( CurvIndex : Integer ) : String
7396: Function GetCurvStep( CurvIndex : Integer ) : Integer
7397: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7398: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7399: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7400: Procedure PlotFunc(Canvas : TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7401: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7402: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7403: Function Xpixel( X : Float ) : Integer
7404: Function Ypixel( Y : Float ) : Integer
7405: Function Xuser( X : Integer ) : Float
7406: Function Yuser( Y : Integer ) : Float
7407: end;
7408:
7409: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7410: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7411: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7412: Procedure FFT_Integer_Cleanup
7413: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7414: //unit uPSI_JclStreams;
7415: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin ) : Int64
7416: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer ) : Int64
7417: Function CompareStreams( A, B : TStream; BufferSize : Integer ) : Boolean

```

```

7418: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7419:
7420: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7421: begin
7422:   FindClass('TOBJECT'),'EInvalidDest
7423:   FindClass('TOBJECT'),'ECantMove
7424:   Procedure fmxCopyFile( const FileName, DestName : string)
7425:   Procedure fmxMoveFile( const FileName, DestName : string)
7426:   Function fmxGetFileSize( const FileName : string) : LongInt
7427:   Function fmxFileDateTime( const FileName : string) : TDateTime
7428:   Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7429:   Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7430: end;
7431:
7432: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7433: begin
7434:   SIRegister_IFindFileIterator(CL);
7435:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7436: end;
7437:
7438: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7439: begin
7440:   Function SkipWhite( cp : PChar) : PChar
7441:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7442:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7443:   Function ReadIdent( cp : PChar; var ident : string) : PChar
7444: end;
7445:
7446: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7447: begin
7448:   SIRegister_TStringHashMapTraits(CL);
7449:   Function CaseSensitiveTraits : TStringHashMapTraits
7450:   Function CaseInsensitiveTraits : TStringHashMapTraits
7451:   THashNode, 'record Str : string; Ptr : Pointer; Left : PHashNod
7452:   +e; Right : PHashNode; end
7453:   //PHashArray', 'THashArray // will not work
7454:   SIRegister_TStringHashMap(CL);
7455:   THashValue, 'Cardinal
7456:   Function StrHash( const s : string) : THashValue
7457:   Function TextHash( const s : string) : THashValue
7458:   Function DataHash( var AValue, ASize : Cardinal) : THashValue
7459:   Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7460:   Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7461:   Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7462:   SIRegister_TCaseSensitiveTraits(CL);
7463:   SIRegister_TCaseInsensitiveTraits(CL);
7464:
7465:
7466: //*****unit uPSI_umath;
7467: Function uExpo( X : Float) : Float
7468: Function uExp2( X : Float) : Float
7469: Function uExp10( X : Float) : Float
7470: Function uLog( X : Float) : Float
7471: Function uLog2( X : Float) : Float
7472: Function uLog10( X : Float) : Float
7473: Function uLogA( X, A : Float) : Float
7474: Function uIntPower( X : Float; N : Integer): Float
7475: Function uPower( X, Y : Float) : Float
7476: Function SgnGamma( X : Float) : Integer
7477: Function Stirling( X : Float) : Float
7478: Function StirLog( X : Float) : Float
7479: Function Gamma( X : Float) : Float
7480: Function LnGamma( X : Float) : Float
7481: Function DiGamma( X : Float) : Float
7482: Function TriGamma( X : Float) : Float
7483: Function IGamma( X : Float) : Float
7484: Function JGamma( X : Float) : Float
7485: Function InvGamma( X : Float) : Float
7486: Function Erf( X : Float) : Float
7487: Function Erfc( X : Float) : Float
7488: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7489: { Correlation coefficient between samples X and Y }
7490: function DBeta(A, B, X : Float) : Float;
7491: { Density of Beta distribution with parameters A and B }
7492: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7493: Function Beta(X, Y : Float) : Float
7494: Function Binomial( N, K : Integer) : Float
7495: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7496: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7497: Procedure LU_Decom( A : TMatrix; Lb, Ub : Integer)
7498: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7499: Function DNorm( X : Float) : Float
7500:
7501: function DGamma(A, B, X : Float) : Float;
7502: { Density of Gamma distribution with parameters A and B }
7503: function DKhi2(Nu : Integer; X : Float) : Float;
7504: { Density of Khi-2 distribution with Nu d.o.f. }
7505: function DStudent(Nu : Integer; X : Float) : Float;
7506: { Density of Student distribution with Nu d.o.f. }

```

```

7507: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7508: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7509: function IBeta(A, B, X : Float) : Float;
7510: { Incomplete Beta function }
7511: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7512:
7513: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7514: begin
7515:   procedure SetOptAlgo( Algo : TOptAlgo)
7516:   procedure SetOptAlgo(Algo : TOptAlgo);
7517:   { -----
7518:     Sets the optimization algorithm according to Algo, which must be
7519:     NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7520:
7521:   Function GetOptAlgo : TOptAlgo
7522:   Procedure SetMaxParam( N : Byte)
7523:   Function GetMaxParam : Byte
7524:   Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7525:   Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7526:   Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7527:   Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7528:   Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7529:   Procedure SetMCFile( FileName : String)
7530:   Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7531:   Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
LastPar:Integer;V:TMatrix);
7532: end;
7533:
7534: (*-----*)
7535: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7536: begin
7537:   procedure SaveSimplex( FileName : string)
7538:   procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7539: end;
7540: (*-----*)
7541: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7542: begin
7543:   procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7544:   procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7545: end;
7546:
7547: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7548: begin
7549:   Function LTrim( S : String) : String
7550:   Function RTrim( S : String) : String
7551:   Function uTrim( S : String) : String
7552:   Function StrChar( N : Byte; C : Char) : String
7553:   Function RFill( S : String; L : Byte) : String
7554:   Function LFill( S : String; L : Byte) : String
7555:   Function CFill( S : String; L : Byte) : String
7556:   Function Replace( S : String; C1, C2 : Char) : String
7557:   Function Extract( S : String; var Index : Byte; Delim : Char) : String
7558:   Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7559:   Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7560:   Function FloatStr( X : Float) : String
7561:   Function IntStr( N : LongInt) : String
7562:   Function uCompStr( Z : Complex) : String
7563: end;
7564:
7565: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7566: begin
7567:   Function uSinh( X : Float) : Float
7568:   Function uCosh( X : Float) : Float
7569:   Function uTanh( X : Float) : Float
7570:   Function uArcSinh( X : Float) : Float
7571:   Function uArcCosh( X : Float) : Float
7572:   Function ArcTanh( X : Float) : Float
7573:   Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7574: end;
7575:
7576: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7577: begin
7578:   type RNG_Type =
7579:   (RNG_MWC,      { Multiply-With-Carry }
7580:   RNG_MT,        { Mersenne Twister }
7581:   RNG_UVAG);    { Universal Virtual Array Generator }
7582:   Procedure SetRNG( RNG : RNG_Type)
7583:   Procedure InitGen( Seed : RNG_IntType)
7584:   Procedure SRand( Seed : RNG_IntType)
7585:   Function IRanGen : RNG_IntType
7586:   Function IRanGen31 : RNG_IntType
7587:   Function RanGen1 : Float
7588:   Function RanGen2 : Float
7589:   Function RanGen3 : Float
7590:   Function RanGen53 : Float
7591: end;
7592:

```



```

7593: // Optimization by Simulated Annealing
7594: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7595: begin
7596:   Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7597:   Procedure SA_CreateLogFile( FileName : String)
7598:   Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7599: end;
7600:
7601: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7602: begin
7603:   Procedure InitUVAGbyString( KeyPhrase : string)
7604:   Procedure InitUVAG( Seed : RNG_IntType)
7605:   Function IRanUVAG : RNG_IntType
7606: end;
7607:
7608: procedure SIRegister_uenalg(CL: TPSPascalCompiler);
7609: begin
7610:   Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7611:   Procedure GA_CreateLogFile( LogFileName : String)
7612:   Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7613: end;
7614:
7615: TVector', 'array of Float
7616: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7617: begin
7618:   Procedure QSort( X : TVector; Lb, Ub : Integer)
7619:   Procedure DQSort( X : TVector; Lb, Ub : Integer)
7620: end;
7621:
7622: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7623: begin
7624:   Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7625:   Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7626: end;
7627:
7628: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7629: begin
7630:   FT_Result', 'Integer
7631:   //TDWordptr', '^DWord // will not work
7632:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor
7633:   d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PChar
7634:   r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP
7635:   ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B
7636:   yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By
7637:   te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte
7638:   ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S
7639:   erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh
7640:   Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By
7641:   te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B
7642:   yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs :
7643:   Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I
7644:   nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv
7645:   ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0
7646:   : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RiSVCP : B
7647:   yte; end
7648: end;
7649:
7650:
7651: //***** PaintFX*****
7652: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7653: begin
7654:   //with RegClassS(CL, 'TComponent', 'TJvPaintFX') do
7655:   with AddClassN(FindClass('TComponent'), 'TJvPaintFX') do begin
7656:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7657:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7658:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7659:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7660:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7661:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7662:     Procedure Turn( Src, Dst : TBitmap)
7663:     Procedure TurnRight( Src, Dst : TBitmap)
7664:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7665:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7666:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7667:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7668:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7669:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7670:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7671:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7672:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7673:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7674:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7675:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7676:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7677:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7678:     Procedure Emboss( var Bmp : TBitmap)
7679:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7680:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7681:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)

```

```

7682: Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7683: Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7684: Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7685: Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7686: Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7687: Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7688: Procedure QuartoOpaque( Src, Dst : TBitmap)
7689: Procedure SemiOpaque( Src, Dst : TBitmap)
7690: Procedure ShadowDownLeft( const Dst : TBitmap)
7691: Procedure ShadowDownRight( const Dst : TBitmap)
7692: Procedure ShadowUpLeft( const Dst : TBitmap)
7693: Procedure ShadowUpRight( const Dst : TBitmap)
7694: Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7695: Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7696: Procedure FlipRight( const Dst : TBitmap)
7697: Procedure FlipDown( const Dst : TBitmap)
7698: Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7699: Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7700: Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7701: Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7702: Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7703: Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7704: Procedure SmoothResize( var Src, Dst : TBitmap)
7705: Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7706: Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7707: Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7708: Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7709: Procedure GrayScale( const Dst : TBitmap)
7710: Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7711: Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7712: Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7713: Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7714: Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7715: Procedure Spray( const Dst : TBitmap; Amount : Integer)
7716: Procedure AntiAlias( const Dst : TBitmap)
7717: Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7718: Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7719: Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7720: Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7721: Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7722: Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7723: Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7724: Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7725: Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7726: Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7727: Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7728: Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7729: Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7730: Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7731: Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7732: Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7733: Procedure Invert( Src : TBitmap)
7734: Procedure MirrorRight( Src : TBitmap)
7735: Procedure MirrorDown( Src : TBitmap)
7736: end;
7737: end;
7738:
7739: (*-----*)
7740: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7741: begin
7742:   AddTypes('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7743:     + 'ye, lbrotate, lbwist, lbwist, lbwist, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7744:     + 'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7745:   );
7746:   SIRegister_TJvPaintFX(CL);
7747:   Function SplineFilter( Value : Single) : Single
7748:   Function BellFilter( Value : Single) : Single
7749:   Function TriangleFilter( Value : Single) : Single
7750:   Function BoxFilter( Value : Single) : Single
7751:   Function HermiteFilter( Value : Single) : Single
7752:   Function Lanczos3Filter( Value : Single) : Single
7753:   Function MitchellFilter( Value : Single) : Single
7754: end;
7755:
7756: (*-----*)
7757: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7758: begin
7759:   TeeMsg_DefaultFunctionName, 'String').SetString( 'TeeFunction
7760:   TeeMsg_DefaultSeriesName, 'String').SetString( 'Series
7761:   TeeMsg_DefaultToolName, 'String').SetString( 'ChartTool
7762:   ChartComponentPalette, 'String').SetString( 'TeeChart
7763:   TeeMaxLegendColumns, 'LongInt').SetInt( 2);
7764:   TeeDefaultLegendSymbolWidth, 'LongInt').SetInt( 20);
7765:   TeeTitleFootDistance, 'LongInt').SetInt( 5);
7766:   SIRegister_TCustomChartWall(CL);
7767:   SIRegister_TChartWall(CL);
7768:   SIRegister_TChartLegendGradient(CL);
7769:   TLegendStyle, '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7770:   TLegendAlignment, '( laLeft, laRight, laTop, laBottom )

```

```

7771: FindClass('TOBJECT'),'LegendException
7772: TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7773:   + 'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7774: FindClass('TOBJECT'),'TCustomChartLegend
7775: TLegendSymbolSize', '( lcsPercent, lcsPixels )
7776: TLegendSymbolPosition', '( spLeft, spRight )
7777: TSymbolDrawEvent', 'Procedure (Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7778: TSymbolCalcHeight', 'Function : Integer
7779: SIRegister_TLegendSymbol(CL);
7780: SIRegister_TTeeCustomShapePosition(CL);
7781: TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7782: SIRegister_TLegendTitle(CL);
7783: SIRegister_TLegendItem(CL);
7784: SIRegister_TLegendItems(CL);
7785: TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7786: FindClass('TOBJECT'),'TCustomChart
7787: SIRegister_TCustomChartLegend(CL);
7788: SIRegister_TChartLegend(CL);
7789: SIRegister_TChartTitle(CL);
7790: SIRegister_TChartFootTitle(CL);
7791: TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7792:   + 'eButton; Shift : TShiftState; X, Y : Integer)
7793: TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7794:   + 'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7795: TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7796:   + 'TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7797: TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7798:   + 'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7799: TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7800: TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7801: TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7802:   + 'toMax : Boolean; Min : Double; Max : Double; end
7803: TAllAxisSavedScales', 'array of TAxisSavedScales
7804: SIRegister_TChartBackWall(CL);
7805: SIRegister_TChartRightWall(CL);
7806: SIRegister_TChartBottomWall(CL);
7807: SIRegister_TChartLeftWall(CL);
7808: SIRegister_TChartWalls(CL);
7809: TChartAllowScrollEvent', 'Procedure (Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7810: SIRegister_TCustomChart(CL);
7811: SIRegister_TChart(CL);
7812: SIRegister_TTeeSeriesTypes(CL);
7813: SIRegister_TTeeToolTypes(CL);
7814: SIRegister_TTeeDragObject(CL);
7815: SIRegister_TColorPalettes(CL);
7816: Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
AGalleryPage:PString;ANumGallerySeries:Integer;
7817: Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7818: Procedure RegisterTeeFunction(AFuncClass:TTeeFunctionClass;ADescription,
AGalleryPage:PString;ANumGallerySeries: Int;
7819: Procedure RegisterTeeBasicFunction( AFuncionClass : TTeeFunctionClass; ADescription : PString)
7820: Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFuncionClass:TTeeFunctionClass;
ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7821: Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7822: Procedure UnRegisterTeeFunctions( const AFuncionList : array of TTeeFunctionClass)
7823: Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7824: Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7825: Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
AFuncionClass : TTeeFunctionClass) : TChartSeries
7826: Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7827: Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7828: Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7829: Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7830: Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7831: Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7832: Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7833: Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7834: Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7835: Function GetGallerySeriesName( ASeries : TChartSeries) : String
7836: Procedure PaintSeriesLegend(ASeries:TChartSeries; ACanvas:TCanvas; const R:TRect;ReferenceChart :
TCustomChart);
7837: SIRegister_TChartTheme(CL);
7838: //TChartThemeClass', 'class of TChartTheme
7839: //TCanvasClass', 'class of TCanvas3D
7840: Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7841: Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7842: Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7843: Procedure ShowMessageUser( const S : String)
7844: Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7845: Function HasLabels( ASeries : TChartSeries) : Boolean
7846: Function HasColors( ASeries : TChartSeries) : Boolean
7847: Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7848: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7849: end;
7850:
7851:
7852: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7853: begin
7854:   //TeeFormBorderStyle', '' ).SetString( bsNone);

```

```

7855: SIRegister_TMetafile(CL);
7856: 'TeeDefVerticalMargin', 'LongInt').SetInt( 4);
7857: 'TeeDefHorizMargin', 'LongInt').SetInt( 3);
7858: 'crTeeHand', 'LongInt').SetInt( TCursor ( 2020 ));
7859: 'TeeMsg_TeeHand', 'String').SetString( 'crTeeHand
7860: 'TeeNormalPrintDetail', 'LongInt').SetInt( 0);
7861: 'TeeHighPrintDetail', 'LongInt').SetInt( - 100);
7862: 'TeeDefault_PrintMargin', 'LongInt').SetInt( 15);
7863: 'MaxDefaultColors', 'LongInt').SetInt( 19);
7864: 'TeeTabDelimiter', 'Char').SetString( #9);
7865: TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7866: + 'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7867: + 'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7868: + 'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7869: + 'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7870: + 'ourMonths, dtSixMonths, dtOneYear, dtNone )
7871: SIRegister_TCustomPanelNoCaption(CL);
7872: FindClass('TObject'), 'TCustomTeePanel
7873: SIRegister_TZoomPanning(CL);
7874: SIRegister_TTeeEvent(CL);
7875: //SIRegister_TTeeEventListeners(CL);
7876: TTeeMouseEventKind', '( meDown, meUp, meMove )
7877: SIRegister_TTeeMouseEvent(CL);
7878: SIRegister_TCustomTeePanel(CL);
7879: //TChartGradient', 'TGradient
7880: //TChartGradientClass', 'class of TChartGradient
7881: TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7882: SIRegister_TTeeZoomPen(CL);
7883: SIRegister_TTeeZoomBrush(CL);
7884: TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7885: SIRegister_TTeeZoom(CL);
7886: FindClass('TObject'), 'TCustomTeePanelExtended
7887: TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7888: SIRegister_TBackImage(CL);
7889: SIRegister_TCustomTeePanelExtended(CL);
7890: //TChartBrushClass', 'class of TChartBrush
7891: SIRegister_TTeeCustomShapeBrushPen(CL);
7892: TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7893: TTextFormat', '( ttfNormal, ttfHtml )
7894: SIRegister_TTeeCustomShape(CL);
7895: SIRegister_TTeeShape(CL);
7896: SIRegister_TTeeExportData(CL);
7897: Function TeeStr( const Num : Integer) : String
7898: Function DateTimeDefaultFormat( const AStep : Double) : String
7899: Function TEEDaysInMonth( Year, Month : Word) : Word
7900: Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7901: Function NextDateTimeStep( const AStep : Double) : Double
7902: Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7903: Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7904: Function PointInLine2( const P, FromPoint, ToPoint : TPoint; const TolerancePixels : Integer) : Boolean;
7905: Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels : Integer) : Boolean;
7906: Function PointInLineTolerance( const P : TPoint; const px, py, qx, qy, TolerancePixels : Integer) : Boolean;
7907: Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7908: Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7909: Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7910: Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7911: Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7912: Function PointInEllipse1( const P : TPoint; Left, Top, Right, Bottom : Integer) : Boolean;
7913: Function DelphiToLocalFormat( const Format : String) : String
7914: Function LocalToDelphiFormat( const Format : String) : String
7915: Procedure TEENableControls( Enable : Boolean; const ControlArray : array of TControl)
7916: Function TeeRoundDate( const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7917: Procedure TeeDateTimeIncrement( IsDateTime : Boolean; Increment : Boolean; var Value : Double; const
AnIncrement : Double; tmpWhichDateTime : TDateTimeStep)
7918: 'TeeSortCompare', 'Function ( a, b : Integer) : Integer
7919: 'TeeSortSwap', 'Procedure ( a, b : Integer)
7920: Procedure TeeSort( StartIndex, EndIndex : Integer; CompareFunc : TTeeSortCompare; SwapFunc : TTeeSortSwap);
7921: Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7922: Function TeeExtractField( St : String; Index : Integer) : String;
7923: Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7924: Function TeeNumFields( St : String) : Integer;
7925: Function TeeNumFields1( const St, Separator : String) : Integer;
7926: Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7927: Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7928: // TColorArray', 'array of TColor
7929: Function GetDefaultColor( const Index : Integer) : TColor
7930: Procedure SetDefaultColorPalette;
7931: Procedure SetDefaultColorPalette1( const Palette : array of TColor);
7932: 'TeeCheckBoxSize', 'LongInt').SetInt( 11);
7933: Procedure TeeDrawCheckBox( x, y : Integer; Canvas : TCanvas; Checked : Boolean; ABackColor : TColor; CheckBox : Boolean);
7934: Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
7935: Function TryStrToFloat( const S : String; var Value : Double) : Boolean
7936: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
7937: Procedure TeeTranslateControl( AControl : TControl);
7938: Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7939: Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
7940: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
7941: Function TeeAntialias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
7942: //Procedure DrawBevel( Canvas : TTeeCanvas; Bevel : TPanelBevel; var R : TRect; Width : Integer; Round : Integer);

```



```

7943: //Function ScreenRatio( ACanvas : TCanvas3D) : Double
7944: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
7945: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
7946: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
7947: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
7948: Function TeeReadStringOption( const AKey, DefaultValue : String) : String
7949: Procedure TeeSaveStringOption( const AKey, Value : String)
7950: Function TeeDefaultXMLEncoding : String
7951: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
7952: TeeWindowHandle', Integer
7953: Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
7954: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
7955: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
7956: end;
7957:
7958:
7959: using mXBDEUtils
7960: *****
7961: Procedure SetAlias( aAlias, aDirectory : String)
7962: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
7963: Function GetFileVersionNumber( const FileName : String) : TVersionNo
7964: Procedure SetBDE( aPath, aNode, aValue : String)
7965: function RestartDialog(Wnd: HWND; Reason: PChar; Flags: Integer): Integer; stdcall;
7966: Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
7967: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
7968: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
7969: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
7970: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
7971:
7972:
7973: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
7974: begin
7975: AddClassN(FindClass('TOBJECT'),'EDateTime
7976: Function DatePart( const D : TDateTime) : Integer
7977: Function TimePart( const D : TDateTime) : Double
7978: Function Century( const D : TDateTime) : Word
7979: Function Year( const D : TDateTime) : Word
7980: Function Month( const D : TDateTime) : Word
7981: Function Day( const D : TDateTime) : Word
7982: Function Hour( const D : TDateTime) : Word
7983: Function Minute( const D : TDateTime) : Word
7984: Function Second( const D : TDateTime) : Word
7985: Function Millisecond( const D : TDateTime) : Word
7986: ('OneDay','Extended').setExtended( 1.0);
7987: ('OneHour','Extended').SetExtended( OneDay / 24);
7988: ('OneMinute','Extended').SetExtended( OneHour / 60);
7989: ('OneSecond','Extended').SetExtended( OneMinute / 60);
7990: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
7991: ('OneWeek','Extended').SetExtended( OneDay * 7);
7992: ('HoursPerDay','Extended').SetExtended( 24);
7993: ('MinutesPerHour','Extended').SetExtended( 60);
7994: ('SecondsPerMinute','Extended').SetExtended( 60);
7995: Procedure SetYear( var D : TDateTime; const Year : Word)
7996: Procedure SetMonth( var D : TDateTime; const Month : Word)
7997: Procedure SetDay( var D : TDateTime; const Day : Word)
7998: Procedure SetHour( var D : TDateTime; const Hour : Word)
7999: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8000: Procedure SetSecond( var D : TDateTime; const Second : Word)
8001: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8002: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8003: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8004: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8005: Function IsAM( const D : TDateTime) : Boolean
8006: Function IsPM( const D : TDateTime) : Boolean
8007: Function IsMidnight( const D : TDateTime) : Boolean
8008: Function IsNoon( const D : TDateTime) : Boolean
8009: Function IsSunday( const D : TDateTime) : Boolean
8010: Function IsMonday( const D : TDateTime) : Boolean
8011: Function IsTuesday( const D : TDateTime) : Boolean
8012: Function IsWednesday( const D : TDateTime) : Boolean
8013: Function IsThursday( const D : TDateTime) : Boolean
8014: Function IsFriday( const D : TDateTime) : Boolean
8015: Function IsSaturday( const D : TDateTime) : Boolean
8016: Function IsWeekend( const D : TDateTime) : Boolean
8017: Function Noon( const D : TDateTime) : TDateTime
8018: Function Midnight( const D : TDateTime) : TDateTime
8019: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8020: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8021: Function NextWorkday( const D : TDateTime) : TDateTime
8022: Function PreviousWorkday( const D : TDateTime) : TDateTime
8023: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8024: Function LastDayOfYear( const D : TDateTime) : TDateTime
8025: Function EasterSunday( const Year : Word) : TDateTime
8026: Function GoodFriday( const Year : Word) : TDateTime
8027: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8028: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8029: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8030: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime

```

```

8031: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8032: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8033: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8034: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8035: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8036: Function DayOfYear( const D : TDateTime) : Integer
8037: Function DaysInMonth( const Ye, Mo : Word) : Integer
8038: Function DaysInMonth( const D : TDateTime) : Integer
8039: Function DaysInYear( const Ye : Word) : Integer
8040: Function DaysInYearDate( const D : TDateTime) : Integer
8041: Function WeekNumber( const D : TDateTime) : Integer
8042: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8043: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8044: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8045: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8046: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8047: Function DiffHours( const D1, D2 : TDateTime) : Integer
8048: Function DiffDays( const D1, D2 : TDateTime) : Integer
8049: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8050: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8051: Function DiffYears( const D1, D2 : TDateTime) : Integer
8052: Function GMTBias : Integer
8053: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8054: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8055: Function NowAsGMTTime : TDateTime
8056: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8057: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8058: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8059: Function DateTimeToANSI( const D : TDateTime) : Integer
8060: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8061: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8062: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8063: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8064: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8065: Function DateTimeAsElapsedTime( const D: TDateTime; const IncludeMilliseconds: Boolean) : AnsiString
8066: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8067: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8068: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8069: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8070: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8071: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8072: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8073: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8074: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8075: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8076: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8077: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8078: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8079: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8080: Function EnglishShortMonthA( const S : AnsiString) : Integer
8081: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8082: Function EnglishLongMonthA( const S : AnsiString) : Integer
8083: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8084: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8085: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8086: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8087: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8088: Function RFCMonthA( const S : AnsiString) : Word
8089: Function RFCMonthU( const S : UnicodeString) : Word
8090: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds: Boolean) : AnsiString
8091: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds: Boolean) : UnicodeString
8092: Function GMTDateTimeToRFC1123DateTimeA( const D: TDateTime; const IncludeDayOfWeek: Bool) : AnsiString;
8093: Function GMTDateTimeToRFC1123DateTimeU( const D: TDateTime; const IncludeDayOfWeek: Bool) : UnicodeString;
8094: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8095: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8096: Function NowAsRFCDateTimeA : AnsiString
8097: Function NowAsRFCDateTimeU : UnicodeString
8098: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8099: Function RFCDateTimeToTime( const S : AnsiString) : TDateTime
8100: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8101: Function TimePeriodStr( const D : TDateTime) : AnsiString
8102: Procedure SelfTest
8103: end;
8104: //*****CFFileUtils
8105: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8106: Function PathHasDriveLetter( const Path : String) : Boolean
8107: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8108: Function PathIsDriveLetter( const Path : String) : Boolean
8109: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8110: Function PathIsDriveRoot( const Path : String) : Boolean
8111: Function PathIsRootA( const Path : AnsiString) : Boolean
8112: Function PathIsRoot( const Path : String) : Boolean
8113: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8114: Function PathIsUNCPath( const Path : String) : Boolean
8115: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8116: Function PathIsAbsolute( const Path : String) : Boolean
8117: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8118: Function PathIsDirectory( const Path : String) : Boolean
8119: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString

```

```

8120: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8121: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8122: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8123: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8124: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8125: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8126: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8127: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8128: Function PathCanonical( const Path : String; const PathSep : Char) : String
8129: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8130: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8131: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8132: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8133: Procedure PathSplitLeftElementA(const Path:ASString;var LeftElement,RightPath:ASString;const PathSep:Char);
8134: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8135: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8136: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8137: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8138: Function FileNameValid( const FileName : String) : String
8139: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8140: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8141: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8142: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8143: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8144: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8145: Procedure CCopyFile( const FileName, DestName : String)
8146: Procedure CMoveFile( const FileName, DestName : String)
8147: Function CDeleteFiles( const FileMask : String) : Boolean
8148: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8149: Procedure FileCloseEx( const FileHandle : TFileHandle)
8150: Function FileExistsA( const FileName : AnsiString) : Boolean
8151: Function CFileExists( const FileName : String) : Boolean
8152: Function CFileSize( const FileName : String) : Int64
8153: Function FileGetDateTime( const FileName : String) : TDateTime
8154: Function FileGetDateTime2( const FileName : String) : TDateTime
8155: Function FileIsReadOnly( const FileName : String) : Boolean
8156: Procedure FileDeleteEx( const FileName : String)
8157: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8158: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
: TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8159: Function DirectoryEntryExists( const Name : String) : Boolean
8160: Function DirectoryEntrySize( const Name : String) : Int64
8161: Function CDirectoryExists( const DirectoryName : String) : Boolean
8162: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8163: Procedure CDirectoryCreate( const DirectoryName : String)
8164: Function GetFirstFileNameMatching( const FileMask : String) : String
8165: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8166: Function DirEntriesDirectory( const FileName : AnsiString) : Boolean
8167: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8168: AddTypeS('LogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8169: + 'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8170: Function DriveIsValid( const Drive : Char) : Boolean
8171: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8172: Function DriveFreeSpace( const Path : AnsiString) : Int64
8173:
8174: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8175: begin
8176:   AddClassN(FindClass('TOBJECT'),'ETimers
8177:   Const('TickFrequency','LongInt').SetInt( 1000);Function GetTick : LongWord
8178:   Function TickDelta( const D1, D2 : LongWord) : Integer
8179:   Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8180:   AddTypeS('THPTimer', 'Int64
8181:   Procedure StartTimer( var Timer : THPTimer)
8182:   Procedure StopTimer( var Timer : THPTimer)
8183:   Procedure ResumeTimer( var StoppedTimer : THPTimer)
8184:   Procedure InitStoppedTimer( var Timer : THPTimer)
8185:   Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8186:   Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8187:   Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8188:   Procedure WaitMicroseconds( const MicroSeconds : Integer)
8189:   Function GetHighPrecisionFrequency : Int64
8190:   Function GetHighPrecisionTimerOverhead : Int64
8191:   Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8192:   Procedure SelfTestCTimer
8193: end;
8194:
8195: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8196: begin
8197:   Function RandomSeed : LongWord
8198:   Procedure AddEntropy( const Value : LongWord)
8199:   Function RandomUniform : LongWord;
8200:   Function RandomUniform1( const N : Integer) : Integer;
8201:   Function RandomBoolean : Boolean
8202:   Function RandomByte : Byte
8203:   Function RandomByteNonZero : Byte
8204:   Function RandomWord : Word
8205:   Function RandomInt64 : Int64;
8206:   Function RandomInt641( const N : Int64) : Int64;
8207:   Function RandomHex( const Digits : Integer) : String

```

```

8208: Function RandomFloat : Extended
8209: Function RandomAlphaStr( const Length : Integer ) : AnsiString
8210: Function RandomPseudoWord( const Length : Integer ) : AnsiString
8211: Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8212: Function mwcRandomLongWord : LongWord
8213: Function urnRandomLongWord : LongWord
8214: Function moaRandomFloat : Extended
8215: Function mwcRandomFloat : Extended
8216: Function RandomNormalF : Extended
8217: Procedure SelfTestCRandom
8218: end;
8219:
8220: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8221: begin
8222:   // PIntArray, 'TIntArray // will not work
8223:   Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8224:   Addtypes('TConvertTabsProcEx','function(const Line:AnsiString; TabWidth: integer;var HasTabs: boolean):
AnsiString
8225:   Function synMax( x, y : integer ) : integer
8226:   Function synMin( x, y : integer ) : integer
8227:   Function synMinMax( x, mi, ma : integer ) : integer
8228:   Procedure synSwapInt( var l, r : integer)
8229:   Function synMaxPoint( const P1, P2 : TPoint ) : TPoint
8230:   Function synMinPoint( const P1, P2 : TPoint ) : TPoint
8231:   //Function synGetIntArray( Count : Cardinal; InitialValue : integer ) : PIntArray
8232:   Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8233:   Function synGetBestConvertTabsProc( TabWidth : integer ) : TConvertTabsProc
8234:   Function synConvertTabs( const Line : AnsiString; TabWidth : integer ) : AnsiString
8235:   Function synGetBestConvertTabsProcEx( TabWidth : integer ) : TConvertTabsProcEx
8236:   Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8237:   Function synGetExpandedLength( const aStr : string; aTabWidth : integer ) : integer
8238:   Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string ) : integer
8239:   Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8240:   Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8241:   Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8242:   TStringType, '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8243:   +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8244:   ('C3_NONSPACING','LongInt').SetInt( 1);
8245:   ('C3_DIACRITIC','LongInt').SetInt( 2);
8246:   ('C3_VOWELMARK','LongInt').SetInt( 4);
8247:   ('C3_SYMBOL','LongInt').SetInt( 8);
8248:   ('C3_KATAKANA','LongWord').SetUInt( $0010);
8249:   ('C3_HIRAGANA','LongWord').SetUInt( $0020);
8250:   ('C3_HALFWIDTH','LongWord').SetUInt( $0040);
8251:   ('C3_FULLWIDTH','LongWord').SetUInt( $0080);
8252:   ('C3_IDEOGRAPH','LongWord').SetUInt( $0100);
8253:   ('C3_KASHIDA','LongWord').SetUInt( $0200);
8254:   ('C3_LEXICAL','LongWord').SetUInt( $0400);
8255:   ('C3_ALPHA','LongWord').SetUInt( $8000);
8256:   ('C3_NOTAPPLICABLE','LongInt').SetInt( 0);
8257:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8258:   Function synStrRScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8259:   Function synIsStringType( Value : Word ) : TStringType
8260:   Function synGetEOL( Line : PChar ) : PChar
8261:   Function synEncodeString( s : string ) : string
8262:   Function synDecodeString( s : string ) : string
8263:   Procedure synFreeAndNil( var Obj: TObject)
8264:   Procedure synAssert( Expr : Boolean)
8265:   Function synLastDelimiter( const Delimiters, S : string ) : Integer
8266:   TReplaceFlag, '( rfReplaceAll, rfIgnoreCase )
8267:   TReplaceFlags, 'set of TReplaceFlag )
8268:   Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8269:   Function synGetRValue( RGBValue : TColor ) : byte
8270:   Function synGetGValue( RGBValue : TColor ) : byte
8271:   Function synGetBValue( RGBValue : TColor ) : byte
8272:   Function synRGB( r, g, b : Byte ) : Cardinal
8273:   // THighlighterAttriProc, 'Function ( Highlighter : TSynCustomHigh'
8274:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8275:   //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer ) : Boolean
8276:   Function synCalcFCS( const ABuf, ABufSize : Cardinal ) : Word
8277:   Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8278: end;
8279:
8280: Function GET_APPCOMMAND_LPARAM( lParam : LPARAM ) : WORD
8281: Function GET_DEVICE_LPARAM( lParam : LPARAM ) : WORD
8282: Function GET_KEYSTATE_LPARAM( lParam : LPARAM ) : WORD
8283:
8284: procedure SIRegister_synutil(CL: TPSPascalCompiler);
8285: begin
8286:   Function STimeZoneBias : integer
8287:   Function TimeZone : string
8288:   Function Rfc822DateTime( t : TDateTime ) : string
8289:   Function CDateTime( t : TDateTime ) : string
8290:   Function SimpleDateTime( t : TDateTime ) : string
8291:   Function AnsiCDateTime( t : TDateTime ) : string
8292:   Function GetMonthNumber( Value : String ) : integer
8293:   Function GetTimeFromStr( Value : string ) : TDateTime

```



```

8294: Function GetDateMDYFromStr( Value : string) : TDateTime
8295: Function DecodeRfcDateTime( Value : string) : TDateTime
8296: Function GetUTTime : TDateTime
8297: Function SetUTTime( Newdt : TDateTime) : Boolean
8298: Function SGetTick : LongWord
8299: Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8300: Function CodeInt( Value : Word) : Ansistring
8301: Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8302: Function CodeLongInt( Value : LongInt) : Ansistring
8303: Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8304: Function DumpStr( const Buffer : Ansistring) : string
8305: Function DumpExStr( const Buffer : Ansistring) : string
8306: Procedure Dump( const Buffer : Ansistring; DumpFile : string)
8307: Procedure DumpEx( const Buffer : Ansistring; DumpFile : string)
8308: Function TrimSPLeft( const S : string) : string
8309: Function TrimSPRight( const S : string) : string
8310: Function TrimSP( const S : string) : string
8311: Function SeparateLeft( const Value, Delimiter : string) : string
8312: Function SeparateRight( const Value, Delimiter : string) : string
8313: Function SGetParameter( const Value, Parameter : string) : string
8314: Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8315: Procedure ParseParameters( Value : string; const Parameters : TStrings)
8316: Function IndexByBegin( Value : string; const List : TStrings) : integer
8317: Function GetEmailAddr( const Value : string) : string
8318: Function GetEmailDesc( Value : string) : string
8319: Function CStrToHex( const Value : Ansistring) : string
8320: Function CIntToBin( Value : Integer; Digits : Byte) : string
8321: Function CBinToInt( const Value : string) : Integer
8322: Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8323: Function CReplaceString( Value, Search, Replace : Ansistring) : Ansistring
8324: Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8325: Function CRPos( const Sub, Value : string) : Integer
8326: Function FetchBin( var Value : string; const Delimiter : string) : string
8327: Function CFetch( var Value : string; const Delimiter : string) : string
8328: Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8329: Function IsBinaryString( const Value : Ansistring) : Boolean
8330: Function PosCRLF( const Value : Ansistring; var Terminator : Ansistring) : integer
8331: Procedure StringsTrim( const value : TStrings)
8332: Function PosFrom( const SubStr, Value : string; From : integer) : integer
8333: Function IncPoint( const p : __pointer; Value : integer) : __pointer
8334: Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8335: Function CCountOfChar( const Value : string; aChr : char) : integer
8336: Function UnquoteStr( const Value : string; Quote : Char) : string
8337: Function QuoteStr( const Value : string; Quote : Char) : string
8338: Procedure HeadersToList( const Value : TStrings)
8339: Procedure ListToHeaders( const Value : TStrings)
8340: Function SwapBytes( Value : integer) : integer
8341: Function ReadStrFromStream( const Stream : TStream; len : integer) : Ansistring
8342: Procedure WriteStrToStream( const Stream : TStream; Value : Ansistring)
8343: Function GetTempFile( const Dir, prefix : Ansistring) : Ansistring
8344: Function CPadString( const Value : Ansistring; len : integer; Pad : AnsiChar) : Ansistring
8345: Function CXorString( Indata1, Indata2 : Ansistring) : Ansistring
8346: Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8347: end;
8348:
8349: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8350: begin
8351:   ('CrcBufSize', 'LongInt').SetInt( 2048);
8352: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8353: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8354: Function Adler32OfFile( FileName : Ansistring) : LongInt
8355: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8356: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8357: Function Crc16OfFile( FileName : Ansistring) : Cardinal
8358: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8359: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8360: Function Crc32OfFile( FileName : Ansistring) : LongInt
8361: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8362: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8363: Function InternetSumOfFile( FileName : Ansistring) : Cardinal
8364: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8365: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8366: Function Kermit16OfFile( FileName : Ansistring) : Cardinal
8367: end;
8368:
8369: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8370: begin
8371:   function CreateOleObject(const ClassName: string): IDispatch;
8372:   function GetActiveOleObject(const ClassName: string): IDispatch;
8373:   function ProgIDToClassID(const ProgID: string): TGUID;
8374:   function ClassIDToProgID(const ClassID: TGUID): string;
8375:   function CreateClassID: string;
8376:   function CreateGUIDString: string;
8377:   function CreateGUIDID: string;
8378:   procedure OleError(ErrorCode: longint)
8379:   procedure OleCheck(Result: HRESULT);
8380: end;
8381:
8382: Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch

```

```

8383: Function xGetActiveOleObject( const ClassName : string) : Variant
8384: //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HRESULT
8385: Function DllCanUnloadNow : HRESULT
8386: Function DllRegisterServer : HRESULT
8387: Function DllUnregisterServer : HRESULT
8388: Function VarFromInterface( Unknown : IUnknown) : Variant
8389: Function VarToInterface( const V : Variant) : IDispatch
8390: Function VarToAutoObject( const V : Variant) : TAutoObject
8391: //Procedure
DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8392: //Procedure DispInvokeError( Status : HRESULT; const ExcepInfo : TExcepInfo)
8393: Procedure OleError( ErrorCode : HRESULT)
8394: Procedure OleCheck( Result : HRESULT)
8395: Function StringToClassID( const S : string) : TCLSID
8396: Function ClassIDToString( const ClassID : TCLSID) : string
8397: Function xProgIDToClassID( const ProgID : string) : TCLSID
8398: Function xClassIDToProgID( const ClassID : TCLSID) : string
8399: Function xWideCompareStr( const S1, S2 : WideString) : Integer
8400: Function xWideSameStr( const S1, S2 : WideString) : Boolean
8401: Function xGUIDToString( const ClassID : TGUID) : string
8402: Function xStringToGUID( const S : string) : TGUID
8403: Function xGetModuleName( Module : HMODULE) : string
8404: Function xAcquireExceptionObject : TObject
8405: Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8406: Function xUtf8Encode( const WS : WideString) : UTF8String
8407: Function xUtf8Decode( const S : UTF8String) : WideString
8408: Function xExcludeTrailingPathDelimiter( const S : string) : string
8409: Function xIncludeTrailingPathDelimiter( const S : string) : string
8410: Function XRTLHandleCOMException : HRESULT
8411: Procedure XRTLCheckArgument( Flag : Boolean)
8412: //Procedure XRTLCheckOutArgument( out Arg)
8413: Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
Connection:Longint);
8414: Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8415: Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
RegisterCookie:Int):HRESULT
8416: Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HRESULT
8417: //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HRESULT
8418: Procedure XRTLEnumActiveObjects( Strings : TStrings)
8419: function XRTLDefaultCategoryManager: IUnknown;
8420: function XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8421: // ICatRegister helper functions
8422: function XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8423: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8424: const CategoryManager: IUnknown = nil): HRESULT;
8425: function XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8426: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8427: const CategoryManager: IUnknown = nil): HRESULT;
8428: function XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8429: const CategoryManager: IUnknown = nil): HRESULT;
8430: function XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8431: const CategoryManager: IUnknown = nil): HRESULT;
8432: // ICatInformation helper functions
8433: function XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8434: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8435: const CategoryManager: IUnknown = nil): HRESULT;
8436: function XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8437: const CategoryManager: IUnknown = nil): HRESULT;
8438: function XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8439: const CategoryManager: IUnknown = nil): HRESULT;
8440: function XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8441: const CategoryManager: IUnknown = nil): HRESULT;
8442: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8443: const ADelete: Boolean = True): WideString;
8444: function XRTLRePos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8445: Function XRTLGetVariantAsString( const Value : Variant) : string
8446: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8447: Function XRTLGetTimeZones : TXRTLTimeZones
8448: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8449: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8450: Function GMTNow : TDateTime
8451: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8452: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8453: Procedure XRTLNotImplemented
8454: Procedure XRTLRaiseError( E : Exception)
8455: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8456:
8457:
8458: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8459: begin
8460: SIRegister_IXRTLValue(CL);
8461: SIRegister_TXRTLValue(CL);
8462: //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8463: AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8464: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8465: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8466: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8467: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8468: Function XRTLValue1( const AValue : Integer) : IXRTLValue;

```

```

8469: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8470: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8471: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8472: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8473: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8474: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8475: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8476: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8477: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8478: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8479: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8480: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8481: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8482: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8483: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8484: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8485: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8486: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8487: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8488: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8489: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8490: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8491: //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8492: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8493: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8494: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8495: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8496: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8497: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8498: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8499: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8500: Function XRTLGetAsObjectDef( const IValue : IXRTLValue; const DefValue : TObject; const
ADetachOwnership : Boolean) : TObject;
8501: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8502: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8503: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8504: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8505: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8506: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8507: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8508: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8509: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8510: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8511: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8512: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8513: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8514: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8515: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8516: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8517: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8518: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8519: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8520: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8521: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8522: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8523: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8524: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8525: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8526: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8527: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8528: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8529: end;
8530:
8531: //*****unit uPSI_GR32;*****
8532:
8533: Function Color32( WinColor : TColor) : TColor32;
8534: Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8535: Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8536: Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8537: Function WinColor( Color32 : TColor32) : TColor
8538: Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8539: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8540: Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8541: Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8542: Function RedComponent( Color32 : TColor32) : Integer
8543: Function GreenComponent( Color32 : TColor32) : Integer
8544: Function BlueComponent( Color32 : TColor32) : Integer
8545: Function AlphaComponent( Color32 : TColor32) : Integer
8546: Function Intensity( Color32 : TColor32) : Integer
8547: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8548: Function HSLtoRGB( H, S, L : Single) : TColor32;
8549: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8550: Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8551: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8552: Function WinPalette( const P : TPalette32) : HPALETTE
8553: Function FloatPoint( X, Y : Single) : TFloatPoint;
8554: Function FloatPoint1( const P : TPoint) : TFloatPoint;
8555: Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8556: Function FixedPoint( X, Y : Integer) : TFixedPoint;

```

```

8557: Function FixedPoint1( X, Y : Single) : TFixedPoint;
8558: Function FixedPoint2( const P : TPoint) : TFixedPoint;
8559: Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8560: AddTypes('TRectRounding', (' rrClosest, rrOutside, rrInside ')
8561: Function MakeRect( const L, T, R, B : Integer) : TRect;
8562: Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8563: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8564: Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8565: Function FixedRect1( const ARect : TRect) : TRect;
8566: Function FixedRect2( const FR : TFloatRect) : TRect;
8567: Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8568: Function FloatRect1( const ARect : TRect) : TFloatRect;
8569: Function FloatRect2( const FXR : TRect) : TFloatRect;
8570: Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8571: Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8572: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8573: Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8574: Function GEqualRect( const R1, R2 : TRect) : Boolean;
8575: Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8576: Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8577: Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8578: Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8579: Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8580: Function IsRectEmpty( const R : TRect) : Boolean;
8581: Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8582: Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8583: Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8584: Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8585: Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8586: Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8587: Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8588: Function MessageBeep( uType : UINT) : BOOL
8589: Function ShowCursor( bShow : BOOL) : Integer
8590: Function SetCursorPos( X, Y : Integer) : BOOL
8591: Function SetCursor( hCursor : HICON) : HCURSOR
8592: Function GetCursorPos( var lpPoint : TPoint) : BOOL
8593: //Function ClipCursor( lpRect : PRect) : BOOL
8594: Function GetClipCursor( var lpRect : TRect) : BOOL
8595: Function GetCursor : HCURSOR
8596: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8597: Function GetCaretBlinkTime : UINT
8598: Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8599: Function DestroyCaret : BOOL
8600: Function HideCaret( hWnd : HWND) : BOOL
8601: Function ShowCaret( hWnd : HWND) : BOOL
8602: Function SetCaretPos( X, Y : Integer) : BOOL
8603: Function GetCaretPos( var lpPoint : TPoint) : BOOL
8604: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8605: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8606: Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8607: Function WindowFromPoint( Point : TPoint) : HWND
8608: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8609:
8610:
8611: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8612: begin
8613:   Function FixedFloor( A : TFixed) : Integer
8614:   Function FixedCeil( A : TFixed) : Integer
8615:   Function FixedMul( A, B : TFixed) : TFixed
8616:   Function FixedDiv( A, B : TFixed) : TFixed
8617:   Function OneOver( Value : TFixed) : TFixed
8618:   Function FixedRound( A : TFixed) : Integer
8619:   Function FixedSqr( Value : TFixed) : TFixed
8620:   Function FixedSqrtLP( Value : TFixed) : TFixed
8621:   Function FixedSqrthP( Value : TFixed) : TFixed
8622:   Function FixedCombine( W, X, Y : TFixed) : TFixed
8623:   Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8624:   Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8625:   Function GRHypot( const X, Y : TFloat) : TFloat;
8626:   Function Hypot1( const X, Y : Integer) : Integer;
8627:   Function FastSqrt( const Value : TFloat) : TFloat
8628:   Function FastSqrtBab1( const Value : TFloat) : TFloat
8629:   Function FastSqrtBab2( const Value : TFloat) : TFloat
8630:   Function FastInvSqrt( const Value : Single) : Single;
8631:   Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8632:   Function GRIsPowerOf2( Value : Integer) : Boolean
8633:   Function PrevPowerOf2( Value : Integer) : Integer
8634:   Function NextPowerOf2( Value : Integer) : Integer
8635:   Function Average( A, B : Integer) : Integer
8636:   Function GRSign( Value : Integer) : Integer
8637:   Function FloatMod( x, y : Double) : Double
8638: end;
8639:
8640: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8641: begin
8642:   Function Clamp( const Value : Integer) : Integer;
8643:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8644:   Function StackAlloc( Size : Integer) : Pointer
8645:   Procedure StackFree( P : Pointer)

```



```

8646: Procedure Swap( var A, B : Pointer);
8647: Procedure Swap1( var A, B : Integer);
8648: Procedure Swap2( var A, B : TFixed);
8649: Procedure Swap3( var A, B : TColor32);
8650: Procedure TestSwap( var A, B : Integer);
8651: Procedure TestSwap1( var A, B : TFixed);
8652: Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8653: Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8654: Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8655: Function Constrain1( const Value, Lo, Hi : Single) : Single;
8656: Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8657: Function GRMin( const A, B, C : Integer) : Integer;
8658: Function GRMax( const A, B, C : Integer) : Integer;
8659: Function Clamp( Value, Max : Integer) : Integer;
8660: Function Clamp1( Value, Min, Max : Integer) : Integer;
8661: Function Wrap( Value, Max : Integer) : Integer;
8662: Function Wrap1( Value, Min, Max : Integer) : Integer;
8663: Function Wrap3( Value, Max : Single) : Single;;
8664: Function WrapPow2( Value, Max : Integer) : Integer;
8665: Function WrapPow21( Value, Min, Max : Integer) : Integer;
8666: Function Mirror( Value, Max : Integer) : Integer;
8667: Function Mirror1( Value, Min, Max : Integer) : Integer;
8668: Function MirrorPow2( Value, Max : Integer) : Integer;
8669: Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8670: Function GetOptimalWrap( Max : Integer) : TWrapProc;
8671: Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8672: Function GetOptimalMirror( Max : Integer) : TWrapProc;
8673: Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8674: Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8675: Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8676: Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8677: Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8678: Function Div255( Value : Cardinal) : Cardinal
8679: Function SAR_4( Value : Integer) : Integer
8680: Function SAR_8( Value : Integer) : Integer
8681: Function SAR_9( Value : Integer) : Integer
8682: Function SAR_11( Value : Integer) : Integer
8683: Function SAR_12( Value : Integer) : Integer
8684: Function SAR_13( Value : Integer) : Integer
8685: Function SAR_14( Value : Integer) : Integer
8686: Function SAR_15( Value : Integer) : Integer
8687: Function SAR_16( Value : Integer) : Integer
8688: Function ColorSwap( WinColor : TColor) : TColor32
8689: end;
8690:
8691: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8692: begin
8693:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8694:   Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8695:   Procedure CopyComponents1(Dst:TCustomBmap32;DstX,
DstY:Integer;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8696:   Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8697:   Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8698:   Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8699:   Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8700:   Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8701:   Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8702:   Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8703:   Function CreateBitmask( Components : TColor32Components) : TColor32
8704:   Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8705:   Procedure
ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8706:   Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizedDst : Boolean)
8707: end;
8708:
8709:
8710: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8711: begin
8712:   AddClassN(FindClass('OBJECT'),'EJclNtfsError
8713:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8714:   Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8715:   Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8716:   Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8717:   Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8718:   Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8719:   Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8720:   Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8721:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8722:   //+ 'tedRangeBuffer; MoreData : Boolean; end
8723:   Function NtfsSetSparse( const FileName : string) : Boolean
8724:   Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Integer) : Boolean
8725:   Function NtfsZeroDataByName( const FileName : string; const First, Last : Integer) : Boolean
8726:   //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Integer;var
Ranges:TNTfsAllocRanges):Boolean;
8727:   //Function NtfsGetAllocRangeEntry( const Ranges : TNTfsAllocRanges;
Index:Integer):TFileAllocatedRangeBuffer
8728:   Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8729:   Function NtfsGetSparse( const FileName : string) : Boolean

```

```

8730: Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8731: Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8732: //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8733: Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8734: Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8735: Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8736: Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8737: Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8738: Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8739: AddTypeS('TopLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8740: Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8741: Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8742: Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8743: Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8744: Function NtfsRequestOpLock( Handle : THandle; Kind : TopLock; Overlapped : TOverlapped) : Boolean
8745: Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8746: Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8747: Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8748: AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8749: + 'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8750: AddTypeS('TStreamIds', 'set of TStreamId
8751: AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8752: + ' : __Pointer; StreamIds : TStreamIds; end
8753: AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8754: + 'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8755: Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8756: Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8757: Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8758: Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8759: AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8760: Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8761: Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
List:TStrings):Bool;
8762: Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8763: Function JclAppInstances : TJclAppInstances;
8764: Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8765: Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8766: Procedure ReadMessageData( const Message : TMessage; var Data : __Pointer; var Size : Integer)
8767: Procedure ReadMessageString( const Message : TMessage; var S : string)
8768: Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8769:
8770:
8771: (*-----*)
8772: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8773: begin
8774: FindClass('TOBJECT'),'EJclGraphicsError
8775: TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8776: TDynPointArray', 'array of TPoint
8777: TDynDynPointArrayArray', 'array of TDynPointArray
8778: TPointF', 'record X : Single; Y : Single; end
8779: TDynPointArrayF', 'array of TPointF
8780: TDrawMode2', '( dmOpaque, dmBlend )
8781: TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8782: TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8783: TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8784: TMatrix3d', 'record array[0..2,0..2] of extended end
8785: TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8786: TScanLine', 'array of Integer
8787: TScanLines', 'array of TScanLine
8788: TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8789: TGradientDirection', '( gdVertical, gdHorizontal )
8790: TPolyFillMode', '( fmAlternate, fmWinding )
8791: TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8792: TJclRegionBitmapMode', '( rmInclude, rmExclude )
8793: TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8794: SIRegister_TJclDesktopCanvas(CL);
8795: FindClass('TOBJECT'),'TJclRegion
8796: SIRegister_TJclRegionInfo(CL);
8797: SIRegister_TJclRegion(CL);
8798: SIRegister_TJclThreadPersistent(CL);
8799: SIRegister_TJclCustomMap(CL);
8800: SIRegister_TJclBitmap32(CL);
8801: SIRegister_TJclByteMap(CL);
8802: SIRegister_TJclTransformation(CL);
8803: SIRegister_TJclLinearTransformation(CL);
8804: Procedure Stretch(NewWidth,
NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8805: Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8806: Procedure DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Integer)
8807: Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8808: Procedure BitmapToJpeg( const FileName : string)
8809: Procedure JpegToBitmap( const FileName : string)
8810: Function ExtractIconCount( const FileName : string) : Integer
8811: Function BitmapToIconJ( Bitmap : HBitmap; cx, cy : Integer) : HICON
8812: Function IconToBitmapJ( Icon : HICON) : HBitmap
8813: Procedure
BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8814: Procedure StretchTransfer(Dst:TJclBitmap32;
DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)

```

```

8815: Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8816: Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8817: Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
TGradientDirection) : Boolean;
8818: Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
RegionBitmapMode:TJclRegionBitmapMode): HRGN
8819: Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8820: Procedure ScreenShot1( bm : TBitmap);
8821: Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8822: Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8823: Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8824: Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8825: Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8826: Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8827: Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8828: Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8829: Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8830: Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8831: Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8832: Procedure Invert( Dst, Src : TJclBitmap32)
8833: Procedure InvertRGB( Dst, Src : TJclBitmap32)
8834: Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8835: Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8836: Procedure SetGamma( Gamma : Single)
8837: end;
8838:
8839: (*-----*)
8840: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8841: begin
8842:   Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8843:   Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8844:   Function LockedCompareExchange1( var Target : __Pointer; Exch, Comp : __Pointer) : Pointer;
8845:   Function LockedDec( var Target : Integer) : Integer
8846:   Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8847:   Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8848:   Function LockedExchangeDec( var Target : Integer) : Integer
8849:   Function LockedExchangeInc( var Target : Integer) : Integer
8850:   Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8851:   Function LockedInc( var Target : Integer) : Integer
8852:   Function LockedSub( var Target : Integer; Value : Integer) : Integer
8853:   TJclWaitResult, '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8854:   SIRegister_TJclDispatcherObject(CL);
8855:   Function WaitForMultipleObjects(const Objects:array of
TJclDispatcherObject;WaitAll:Bool;Timeout:Cardinal):Cardinal;
8856:   Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
Timeout : Cardinal):Cardinal
8857:   SIRegister_TJclCriticalSection(CL);
8858:   SIRegister_TJclCriticalSectionEx(CL);
8859:   SIRegister_TJclEvent(CL);
8860:   SIRegister_TJclWaitableTimer(CL);
8861:   SIRegister_TJclSemaphore(CL);
8862:   SIRegister_TJclMutex(CL);
8863:   POptexSharedInfo, '^TOptexSharedInfo // will not work
8864:   TOptexSharedInfo, 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8865:   SIRegister_TJclOptex(CL);
8866:   TMrewPreferred, '( mpReaders, mpWriters, mpEqual )
8867:   TMrewThreadInfo, 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8868:   TMrewThreadInfoArray, 'array of TMrewThreadInfo
8869:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8870:   PMetSectSharedInfo, '^TMetSectSharedInfo // will not work
8871:   TMetSectSharedInfo, 'record Initialized : LongBool; SpinLock : '
+ 'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8872:   PMeteredSection, '^TMeteredSection // will not work
8873:   TMeteredSection, 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8874:   SIRegister_TJclMeteredSection(CL);
8875:   TEventInfo, 'record EventType : Longint; Signaled : LongBool; end
8876:   TMutexInfo, 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8877:   TSemaphoreCounts, 'record CurrentCount : Longint; MaximumCount : Longint; end
8878:   TTimerInfo, 'record Remaining : TLargeInteger; Signaled : LongBool; end
8879:   Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8880:   Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8881:   Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8882:   Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8883:   Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8884:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8885:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8886:   FindClass('TOBJECT'),'EJclCriticalSectionError
8887:   FindClass('TOBJECT'),'EJclEventError
8888:   FindClass('TOBJECT'),'EJclWaitableTimerError
8889:   FindClass('TOBJECT'),'EJclSemaphoreError
8890:   FindClass('TOBJECT'),'EJclMutexError
8891:   FindClass('TOBJECT'),'EJclMeteredSectionError
8892: end;
8893:
8894:
8895:
8896: //*****unit uPSI_mORMotReport;
8897: Procedure SetCurrentPrinterAsDefault
8898: Function CurrentPrinterName : string
8899: Function mCurrentPrinterPaperSize : string

```

```

8900: Procedure UseDefaultPrinter
8901:
8902: procedure SIRegisterTSTREAM(C1: TPSPascalCompiler);
8903: begin
8904:   with FindClass('TOBJECT'), 'TStream' do begin
8905:     IsAbstract := True;
8906:     //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8907:     // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8908:     function Read(Buffer: String; Count: LongInt): LongInt
8909:     function Write(Buffer: String; Count: LongInt): LongInt
8910:     function ReadString(Buffer: String; Count: LongInt): LongInt //FileStream
8911:     function WriteString(Buffer: String; Count: LongInt): LongInt
8912:     function ReadInt(Buffer: integer; Count: LongInt): LongInt
8913:     function WriteInt(Buffer: integer; Count: LongInt): LongInt
8914:     procedure ReadAB(Buffer: TByteArray; Count: LongInt)
8915:     procedure WriteAB(Buffer: TByteArray; Count: LongInt)
8916:     procedure ReadABD(Buffer: TByteDynArray; Count: LongInt)
8917:     procedure WriteABD(Buffer: TByteDynArray; Count: LongInt)
8918:     procedure ReadAC(Buffer: TCharArray; Count: LongInt)
8919:     procedure WriteAC(Buffer: TCharArray; Count: LongInt)
8920:     procedure ReadACD(Buffer: TCharDynArray; Count: LongInt)
8921:     procedure WriteACD(Buffer: TCharDynArray; Count: LongInt)
8922:
8923:     function Seek(Offset: LongInt; Origin: Word): LongInt
8924:     procedure ReadBuffer(Buffer: String; Count: LongInt)
8925:     procedure WriteBuffer(Buffer: String; Count: LongInt)
8926:     procedure ReadBufferInt(Buffer: Integer; Count: LongInt)';
8927:     procedure WriteBufferInt(Buffer: Integer; Count: LongInt)';
8928:     procedure ReadBufferFloat(Buffer: Double; Count: LongInt)';
8929:     Procedure WriteBufferFloat(Buffer: Double; Count: LongInt)';
8930:
8931:     procedure ReadBufferAB(Buffer: TByteArray; Count: LongInt)
8932:     procedure WriteBufferAB(Buffer: TByteArray; Count: LongInt)
8933:     procedure ReadBufferABD(Buffer: TByteDynArray; Count: LongInt)
8934:     procedure WriteBufferABD(Buffer: TByteDynArray; Count: LongInt)
8935:     procedure ReadBufferAW(Buffer: TWordArray; Count: LongInt)
8936:     procedure WriteBufferAW(Buffer: TWordArray; Count: LongInt)
8937:     procedure ReadBufferAWD(Buffer: TWordDynArray; Count: LongInt)
8938:     procedure WriteBufferAWD(Buffer: TWordDynArray; Count: LongInt)
8939:     procedure ReadBufferAW(Buffer: TWordArray; Count: LongInt)
8940:     procedure WriteBufferAW(Buffer: TWordArray; Count: LongInt)
8941:     procedure ReadBufferAC(Buffer: TCharArray; Count: LongInt)
8942:     procedure WriteBufferAC(Buffer: TCharArray; Count: LongInt)
8943:     procedure ReadBufferACD(Buffer: TCharDynArray; Count: LongInt)
8944:     procedure WriteBufferACD(Buffer: TCharDynArray; Count: LongInt)
8945:
8946:     procedure ReadBufferP(Buffer: PChar; Count: LongInt)
8947:     procedure WriteBufferP(Buffer: PChar; Count: LongInt)
8948:     procedure ReadBufferO(Buffer: TObject; Count: LongInt)';
8949:     procedure WriteBufferO(Buffer: TObject; Count: LongInt)';
8950:     //READBUFFERAC
8951:     function InstanceSize: Longint
8952:     Procedure FixupResourceHeader( FixupInfo : Integer)
8953:     Procedure ReadResHeader
8954:
8955:     {$IFDEF DELPHI4UP}
8956:     function CopyFrom(Source: TStream; Count: Int64): LongInt
8957:     {$ELSE}
8958:     function CopyFrom(Source: TStream; Count: Integer): LongInt
8959:     {$ENDIF}
8960:     RegisterProperty('Position', 'LongInt', iptrw);
8961:     RegisterProperty('Size', 'LongInt', iptrw);
8962:   end;
8963: end;
8964:
8965:
8966: { *****
8967:   Unit DMATH - Interface for DMATH.DLL
8968:   ***** }
8969: // see more docs/dmath_manual.pdf
8970:
8971: Function InitEval : Integer
8972: Procedure SetVariable( VarName : Char; Value : Float)
8973: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
8974: Function Eval( ExpressionString : String) : Float
8975:
8976: unit dmath; //types are in built, others are external in DLL
8977: interface
8978: {$IFDEF DELPHI}
8979: uses
8980:   StdCtrls, Graphics;
8981: {$ENDIF}
8982: { -----
8983:   Types and constants
8984:   ----- }
8985: {$i types.inc}
8986: { -----
8987:   Error handling
8988:   ----- }

```



```

8989: procedure SetErrCode(ErrCode : Integer); external 'dmath';
8990: { Sets the error code }
8991: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
8992: { Sets error code and default function value }
8993: function MathErr : Integer; external 'dmath';
8994: { Returns the error code }
8995: { -----
8996:   Dynamic arrays
8997: ----- }
8998: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
8999: { Sets the auto-initialization of arrays }
9000: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9001: { Creates floating point vector V[0..Ub] }
9002: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9003: { Creates integer vector V[0..Ub] }
9004: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9005: { Creates complex vector V[0..Ub] }
9006: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9007: { Creates boolean vector V[0..Ub] }
9008: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9009: { Creates string vector V[0..Ub] }
9010: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9011: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9012: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9013: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9014: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9015: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9016: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9017: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9018: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9019: { Creates string matrix A[0..Ub1, 0..Ub2] }
9020: { -----
9021:   Minimum, maximum, sign and exchange
9022: ----- }
9023: function FMin(X, Y : Float) : Float; external 'dmath';
9024: { Minimum of 2 reals }
9025: function FMax(X, Y : Float) : Float; external 'dmath';
9026: { Maximum of 2 reals }
9027: function IMin(X, Y : Integer) : Integer; external 'dmath';
9028: { Minimum of 2 integers }
9029: function IMax(X, Y : Integer) : Integer; external 'dmath';
9030: { Maximum of 2 integers }
9031: function Sgn(X : Float) : Integer; external 'dmath';
9032: { Sign (returns 1 if X = 0) }
9033: function Sgn0(X : Float) : Integer; external 'dmath';
9034: { Sign (returns 0 if X = 0) }
9035: function DSgn(A, B : Float) : Float; external 'dmath';
9036: { Sgn(B) * |A| }
9037: procedure FSwap(var X, Y : Float); external 'dmath';
9038: { Exchange 2 reals }
9039: procedure ISwap(var X, Y : Integer); external 'dmath';
9040: { Exchange 2 integers }
9041: { -----
9042:   Rounding functions
9043: ----- }
9044: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9045: { Rounds X to N decimal places }
9046: function Ceil(X : Float) : Integer; external 'dmath';
9047: { Ceiling function }
9048: function Floor(X : Float) : Integer; external 'dmath';
9049: { Floor function }
9050: { -----
9051:   Logarithms, exponentials and power
9052: ----- }
9053: function Expo(X : Float) : Float; external 'dmath';
9054: { Exponential }
9055: function Exp2(X : Float) : Float; external 'dmath';
9056: { 2^X }
9057: function Exp10(X : Float) : Float; external 'dmath';
9058: { 10^X }
9059: function Log(X : Float) : Float; external 'dmath';
9060: { Natural log }
9061: function Log2(X : Float) : Float; external 'dmath';
9062: { Log, base 2 }
9063: function Log10(X : Float) : Float; external 'dmath';
9064: { Decimal log }
9065: function LogA(X, A : Float) : Float; external 'dmath';
9066: { Log, base A }
9067: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9068: { X^N }
9069: function Power(X, Y : Float) : Float; external 'dmath';
9070: { X^Y, X >= 0 }
9071: { -----
9072:   Trigonometric functions
9073: ----- }
9074: function Pythag(X, Y : Float) : Float; external 'dmath';
9075: { Sqrt(X^2 + Y^2) }
9076: function FixAngle(Theta : Float) : Float; external 'dmath';
9077: { Set Theta in -Pi..Pi }

```

```

9078: function Tan(X : Float) : Float; external 'dmath';
9079: { Tangent }
9080: function ArcSin(X : Float) : Float; external 'dmath';
9081: { Arc sinus }
9082: function ArcCos(X : Float) : Float; external 'dmath';
9083: { Arc cosinus }
9084: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9085: { Angle (Ox, OM) with M(X,Y) }
9086: { -----
9087:   Hyperbolic functions
9088:   ----- }
9089: function Sinh(X : Float) : Float; external 'dmath';
9090: { Hyperbolic sine }
9091: function Cosh(X : Float) : Float; external 'dmath';
9092: { Hyperbolic cosine }
9093: function Tanh(X : Float) : Float; external 'dmath';
9094: { Hyperbolic tangent }
9095: function ArcSinh(X : Float) : Float; external 'dmath';
9096: { Inverse hyperbolic sine }
9097: function ArcCosh(X : Float) : Float; external 'dmath';
9098: { Inverse hyperbolic cosine }
9099: function ArcTanh(X : Float) : Float; external 'dmath';
9100: { Inverse hyperbolic tangent }
9101: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9102: { Sinh & Cosh }
9103: { -----
9104:   Gamma function and related functions
9105:   ----- }
9106: function Fact(N : Integer) : Float; external 'dmath';
9107: { Factorial }
9108: function SgnGamma(X : Float) : Integer; external 'dmath';
9109: { Sign of Gamma function }
9110: function Gamma(X : Float) : Float; external 'dmath';
9111: { Gamma function }
9112: function LnGamma(X : Float) : Float; external 'dmath';
9113: { Logarithm of Gamma function }
9114: function Stirling(X : Float) : Float; external 'dmath';
9115: { Stirling's formula for the Gamma function }
9116: function StirLog(X : Float) : Float; external 'dmath';
9117: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9118: function DiGamma(X : Float) : Float; external 'dmath';
9119: { Digamma function }
9120: function TriGamma(X : Float) : Float; external 'dmath';
9121: { Trigamma function }
9122: function IGamma(A, X : Float) : Float; external 'dmath';
9123: { Incomplete Gamma function }
9124: function JGamma(A, X : Float) : Float; external 'dmath';
9125: { Complement of incomplete Gamma function }
9126: function InvGamma(A, P : Float) : Float; external 'dmath';
9127: { Inverse of incomplete Gamma function }
9128: function Erf(X : Float) : Float; external 'dmath';
9129: { Error function }
9130: function Erfc(X : Float) : Float; external 'dmath';
9131: { Complement of error function }
9132: { -----
9133:   Beta function and related functions
9134:   ----- }
9135: function Beta(X, Y : Float) : Float; external 'dmath';
9136: { Beta function }
9137: function IBeta(A, B, X : Float) : Float; external 'dmath';
9138: { Incomplete Beta function }
9139: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9140: { Inverse of incomplete Beta function }
9141: { -----
9142:   Lambert's function
9143:   ----- }
9144: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9145: { -----
9146:   Binomial distribution
9147:   ----- }
9148: function Binomial(N, K : Integer) : Float; external 'dmath';
9149: { Binomial coefficient C(N,K) }
9150: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9151: { Probability of binomial distribution }
9152: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9153: { Cumulative probability for binomial distrib. }
9154: { -----
9155:   Poisson distribution
9156:   ----- }
9157: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9158: { Probability of Poisson distribution }
9159: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9160: { Cumulative probability for Poisson distrib. }
9161: { -----
9162:   Exponential distribution
9163:   ----- }
9164: function DExpo(A, X : Float) : Float; external 'dmath';
9165: { Density of exponential distribution with parameter A }
9166: function FExpo(A, X : Float) : Float; external 'dmath';

```

```

9167: { Cumulative probability function for exponential dist. with parameter A }
9168: { ----- }
9169:   Standard normal distribution
9170:   ----- }
9171: function DNorm(X : Float) : Float; external 'dmath';
9172: { Density of standard normal distribution }
9173: function FNorm(X : Float) : Float; external 'dmath';
9174: { Cumulative probability for standard normal distrib. }
9175: function PNorm(X : Float) : Float; external 'dmath';
9176: { Prob(|U| > X) for standard normal distrib. }
9177: function InvNorm(P : Float) : Float; external 'dmath';
9178: { Inverse of standard normal distribution }
9179: { ----- }
9180:   Student's distribution
9181:   ----- }
9182: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9183: { Density of Student distribution with Nu d.o.f. }
9184: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9185: { Cumulative probability for Student distrib. with Nu d.o.f. }
9186: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9187: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9188: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9189: { Inverse of Student's t-distribution function }
9190: { ----- }
9191:   Khi-2 distribution
9192:   ----- }
9193: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9194: { Density of Khi-2 distribution with Nu d.o.f. }
9195: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9196: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9197: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9198: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9199: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9200: { Inverse of Khi-2 distribution function }
9201: { ----- }
9202:   Fisher-Snedecor distribution
9203:   ----- }
9204: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9205: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9206: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9207: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9208: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9209: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9210: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9211: { Inverse of Snedecor's F-distribution function }
9212: { ----- }
9213:   Beta distribution
9214:   ----- }
9215: function DBeta(A, B, X : Float) : Float; external 'dmath';
9216: { Density of Beta distribution with parameters A and B }
9217: function FBeta(A, B, X : Float) : Float; external 'dmath';
9218: { Cumulative probability for Beta distrib. with param. A and B }
9219: { ----- }
9220:   Gamma distribution
9221:   ----- }
9222: function DGamma(A, B, X : Float) : Float; external 'dmath';
9223: { Density of Gamma distribution with parameters A and B }
9224: function FGamma(A, B, X : Float) : Float; external 'dmath';
9225: { Cumulative probability for Gamma distrib. with param. A and B }
9226: { ----- }
9227:   Expression evaluation
9228:   ----- }
9229: function InitEval : Integer; external 'dmath';
9230: { Initializes built-in functions and returns their number }
9231: function Eval(ExpressionString : String) : Float; external 'dmath';
9232: { Evaluates an expression at run-time }
9233: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9234: { Assigns a value to a variable }
9235: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9236: { Adds a function to the parser }
9237: { ----- }
9238:   Matrices and linear equations
9239:   ----- }
9240: procedure GaussJordan(A          : TMatrix;
9241:                      Lb, Ub1, Ub2 : Integer;
9242:                      var Det      : Float); external 'dmath';
9243: { Transforms a matrix according to the Gauss-Jordan method }
9244: procedure LinEq(A          : TMatrix;
9245:                B          : TVector;
9246:                Lb, Ub     : Integer;
9247:                var Det     : Float); external 'dmath';
9248: { Solves a linear system according to the Gauss-Jordan method }
9249: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9250: { Cholesky factorization of a positive definite symmetric matrix }
9251: procedure LU_Decomposition(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9252: { LU decomposition }
9253: procedure LU_Solve(A          : TMatrix;
9254:                   B          : TVector;
9255:                   Lb, Ub     : Integer;

```

```

9256:           X           : TVector); external 'dmath';
9257: { Solution of linear system from LU decomposition }
9258: procedure QR_Decomp(A       : TMatrix;
9259:                    Lb, Ub1, Ub2 : Integer;
9260:                    R         : TMatrix); external 'dmath';
9261: { QR decomposition }
9262: procedure QR_Solve(Q, R     : TMatrix;
9263:                   B        : TVector;
9264:                   Lb, Ub1, Ub2 : Integer;
9265:                   X          : TVector); external 'dmath';
9266: { Solution of linear system from QR decomposition }
9267: procedure SV_Decomp(A       : TMatrix;
9268:                    Lb, Ub1, Ub2 : Integer;
9269:                    S          : TVector;
9270:                    V          : TMatrix); external 'dmath';
9271: { Singular value decomposition }
9272: procedure SV_SetZero(S      : TVector;
9273:                    Lb, Ub  : Integer;
9274:                    Tol     : Float); external 'dmath';
9275: { Set lowest singular values to zero }
9276: procedure SV_Solve(U       : TMatrix;
9277:                   S        : TVector;
9278:                   V        : TMatrix;
9279:                   B        : TVector;
9280:                   Lb, Ub1, Ub2 : Integer;
9281:                   X         : TVector); external 'dmath';
9282: { Solution of linear system from SVD }
9283: procedure SV_Approx(U      : TMatrix;
9284:                   S       : TVector;
9285:                   V       : TMatrix;
9286:                   Lb, Ub1, Ub2 : Integer;
9287:                   A        : TMatrix); external 'dmath';
9288: { Matrix approximation from SVD }
9289: procedure EigenVals(A      : TMatrix;
9290:                   Lb, Ub  : Integer;
9291:                   Lambda  : TCompVector); external 'dmath';
9292: { Eigenvalues of a general square matrix }
9293: procedure EigenVect(A      : TMatrix;
9294:                   Lb, Ub  : Integer;
9295:                   Lambda  : TCompVector;
9296:                   V       : TMatrix); external 'dmath';
9297: { Eigenvalues and eigenvectors of a general square matrix }
9298: procedure EigenSym(A      : TMatrix;
9299:                   Lb, Ub  : Integer;
9300:                   Lambda  : TVector;
9301:                   V       : TMatrix); external 'dmath';
9302: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9303: procedure Jacobi(A      : TMatrix;
9304:                 Lb, Ub, MaxIter : Integer;
9305:                 Tol           : Float;
9306:                 Lambda       : TVector;
9307:                 V            : TMatrix); external 'dmath';
9308: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9309: { -----
9310:   Optimization
9311: ----- }
9312: procedure MinBrack(Func      : TFunc;
9313:                   var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9314: { Brackets a minimum of a function }
9315: procedure GoldSearch(Func      : TFunc;
9316:                     A, B      : Float;
9317:                     MaxIter   : Integer;
9318:                     Tol       : Float;
9319:                     var Xmin, Ymin : Float); external 'dmath';
9320: { Minimization of a function of one variable (golden search) }
9321: procedure LinMin(Func      : TFuncNVar;
9322:                 X, DeltaX  : TVector;
9323:                 Lb, Ub    : Integer;
9324:                 var R      : Float;
9325:                 MaxIter   : Integer;
9326:                 Tol       : Float;
9327:                 var F_min  : Float); external 'dmath';
9328: { Minimization of a function of several variables along a line }
9329: procedure Newton(Func      : TFuncNVar;
9330:                 HessGrad   : THessGrad;
9331:                 X          : TVector;
9332:                 Lb, Ub    : Integer;
9333:                 MaxIter   : Integer;
9334:                 Tol       : Float;
9335:                 var F_min  : Float;
9336:                 G         : TVector;
9337:                 H_inv     : TMatrix;
9338:                 var Det   : Float); external 'dmath';
9339: { Minimization of a function of several variables (Newton's method) }
9340: procedure SaveNewton(FileName : string); external 'dmath';
9341: { Save Newton iterations in a file }
9342: procedure Marquardt(Func      : TFuncNVar;
9343:                   HessGrad   : THessGrad;
9344:                   X          : TVector;

```



```

9345:         Lb, Ub      : Integer;
9346:         MaxIter     : Integer;
9347:         Tol          : Float;
9348:         var F_min    : Float;
9349:         G            : TVector;
9350:         H_inv        : TMatrix;
9351:         var Det       : Float; external 'dmath';
9352: { Minimization of a function of several variables (Marquardt's method) }
9353: procedure SaveMarquardt(FileName : string); external 'dmath';
9354: { Save Marquardt iterations in a file }
9355: procedure BFGS(Func      : TFuncNVar;
9356:               Gradient  : TGradient;
9357:               X          : TVector;
9358:               Lb, Ub     : Integer;
9359:               MaxIter    : Integer;
9360:               Tol        : Float;
9361:               var F_min  : Float;
9362:               G          : TVector;
9363:               H_inv      : TMatrix); external 'dmath';
9364: { Minimization of a function of several variables (BFGS method) }
9365: procedure SaveBFGS(FileName : string); external 'dmath';
9366: { Save BFGS iterations in a file }
9367: procedure Simplex(Func      : TFuncNVar;
9368:                  X          : TVector;
9369:                  Lb, Ub     : Integer;
9370:                  MaxIter    : Integer;
9371:                  Tol        : Float;
9372:                  var F_min  : Float); external 'dmath';
9373: { Minimization of a function of several variables (Simplex) }
9374: procedure SaveSimplex(FileName : string); external 'dmath';
9375: { Save Simplex iterations in a file }
9376: { -----
9377:   Nonlinear equations
9378: ----- }
9379: procedure RootBrack(Func      : TFunc;
9380:                   var X, Y, FX, FY : Float); external 'dmath';
9381: { Brackets a root of function Func between X and Y }
9382: procedure Bisect(Func      : TFunc;
9383:                 var X, Y : Float;
9384:                 MaxIter  : Integer;
9385:                 Tol      : Float;
9386:                 var F     : Float); external 'dmath';
9387: { Bisection method }
9388: procedure Secant(Func      : TFunc;
9389:                 var X, Y : Float;
9390:                 MaxIter  : Integer;
9391:                 Tol      : Float;
9392:                 var F     : Float); external 'dmath';
9393: { Secant method }
9394: procedure NewtEq(Func, Deriv : TFunc;
9395:                 var X      : Float;
9396:                 MaxIter    : Integer;
9397:                 Tol        : Float;
9398:                 var F      : Float); external 'dmath';
9399: { Newton-Raphson method for a single nonlinear equation }
9400: procedure NewtEqs(Equations : TEquations;
9401:                 Jacobian    : TJacobian;
9402:                 X, F        : TVector;
9403:                 Lb, Ub     : Integer;
9404:                 MaxIter    : Integer;
9405:                 Tol        : Float); external 'dmath';
9406: { Newton-Raphson method for a system of nonlinear equations }
9407: procedure Broyden(Equations : TEquations;
9408:                 X, F        : TVector;
9409:                 Lb, Ub     : Integer;
9410:                 MaxIter    : Integer;
9411:                 Tol        : Float); external 'dmath';
9412: { Broyden's method for a system of nonlinear equations }
9413: { -----
9414:   Polynomials and rational fractions
9415: ----- }
9416: function Poly(X      : Float;
9417:              Coef    : TVector;
9418:              Deg      : Integer) : Float; external 'dmath';
9419: { Evaluates a polynomial }
9420: function RFrac(X      : Float;
9421:              Coef      : TVector;
9422:              Deg1, Deg2 : Integer) : Float; external 'dmath';
9423: { Evaluates a rational fraction }
9424: function RootPoll(A, B : Float;
9425:                 var X : Float) : Integer; external 'dmath';
9426: { Solves the linear equation A + B * X = 0 }
9427: function RootPol2(Coef : TVector;
9428:                 Z      : TCompVector) : Integer; external 'dmath';
9429: { Solves a quadratic equation }
9430: function RootPol3(Coef : TVector;
9431:                 Z      : TCompVector) : Integer; external 'dmath';
9432: { Solves a cubic equation }
9433: function RootPol4(Coef : TVector;

```

```

9434:      Z      : TCompVector) : Integer; external 'dmath';
9435: { Solves a quartic equation }
9436: function RootPol(Coef : TVector;
9437:      Deg      : Integer;
9438:      Z      : TCompVector) : Integer; external 'dmath';
9439: { Solves a polynomial equation }
9440: function SetRealRoots(Deg : Integer;
9441:      Z      : TCompVector;
9442:      Tol : Float) : Integer; external 'dmath';
9443: { Set the imaginary part of a root to zero }
9444: procedure SortRoots(Deg : Integer;
9445:      Z      : TCompVector); external 'dmath';
9446: { Sorts the roots of a polynomial }
9447: { -----
9448:   Numerical integration and differential equations
9449:   ----- }
9450: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9451: { Integration by trapezoidal rule }
9452: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9453: { Integral from A to B }
9454: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9455: { Integral from 0 to B }
9456: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9457: { Convolution product at time T }
9458: procedure ConvTrap(Func1, Func2: TFunc; T, Y: TVector; N: Integer); external 'dmath';
9459: { Convolution by trapezoidal rule }
9460: procedure RKF45(F      : TDiffEqs;
9461:      Neqn      : Integer;
9462:      Y, Yp     : TVector;
9463:      var T     : Float;
9464:      Tout, RelErr, AbsErr : Float;
9465:      var Flag   : Integer); external 'dmath';
9466: { Integration of a system of differential equations }
9467: { -----
9468:   Fast Fourier Transform
9469:   ----- }
9470: procedure FFT(NumSamples      : Integer;
9471:      InArray, OutArray : TCompVector); external 'dmath';
9472: { Fast Fourier Transform }
9473: procedure IFFT(NumSamples      : Integer;
9474:      InArray, OutArray : TCompVector); external 'dmath';
9475: { Inverse Fast Fourier Transform }
9476: procedure FFT_Integer(NumSamples      : Integer;
9477:      RealIn, ImagIn : TIntVector;
9478:      OutArray      : TCompVector); external 'dmath';
9479: { Fast Fourier Transform for integer data }
9480: procedure FFT_Integer_Cleanup; external 'dmath';
9481: { Clear memory after a call to FFT_Integer }
9482: procedure CalcFrequency(NumSamples,
9483:      FrequencyIndex : Integer;
9484:      InArray      : TCompVector;
9485:      var FFT      : Complex); external 'dmath';
9486: { Direct computation of Fourier transform }
9487: { -----
9488:   Random numbers
9489:   ----- }
9490: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9491: { Select generator }
9492: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9493: { Initialize generator }
9494: function IRanGen : RNG_IntType; external 'dmath';
9495: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9496: function IRanGen31 : RNG_IntType; external 'dmath';
9497: { 31-bit random integer in [0 .. 2^31 - 1] }
9498: function RanGen1 : Float; external 'dmath';
9499: { 32-bit random real in [0,1] }
9500: function RanGen2 : Float; external 'dmath';
9501: { 32-bit random real in [0,1] }
9502: function RanGen3 : Float; external 'dmath';
9503: { 32-bit random real in (0,1) }
9504: function RanGen53 : Float; external 'dmath';
9505: { 53-bit random real in [0,1] }
9506: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9507: { Initializes the 'Multiply with carry' random number generator }
9508: function IRanMWC : RNG_IntType; external 'dmath';
9509: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9510: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9511: { Initializes Mersenne Twister generator with a seed }
9512: procedure InitMTbyArray(InitKey : array of RNG_LongType;
9513:      KeyLength : Word); external 'dmath';
9514: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9515: function IRanMT : RNG_IntType; external 'dmath';
9516: { Random integer from MT generator }
9517: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9518: { Initializes the UVAG generator with a string }
9519: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9520: { Initializes the UVAG generator with an integer }
9521: function IRanUVAG : RNG_IntType; external 'dmath';
9522: { Random integer from UVAG generator }

```

```

9523: function RanGaussStd : Float; external 'dmath';
9524: { Random number from standard normal distribution }
9525: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9526: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9527: procedure RanMult(M      : TVector; L      : TMatrix;
9528:                   Lb, Ub : Integer;
9529:                   X      : TVector); external 'dmath';
9530: { Random vector from multinormal distribution (correlated) }
9531: procedure RanMultIndep(M, S : TVector;
9532:                       Lb, Ub : Integer;
9533:                       X      : TVector); external 'dmath';
9534: { Random vector from multinormal distribution (uncorrelated) }
9535: procedure InitMHPParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9536: { Initializes Metropolis-Hastings parameters }
9537: procedure GetMHPParams(var NCycles, MaxSim, SavedSim: Integer); external 'dmath';
9538: { Returns Metropolis-Hastings parameters }
9539: procedure Hastings(Func      : TFuncNVar;
9540:                   T         : Float;
9541:                   X         : TVector;
9542:                   V         : TMatrix;
9543:                   Lb, Ub    : Integer;
9544:                   Xmat      : TMatrix;
9545:                   X_min     : TVector;
9546:                   var F_min : Float); external 'dmath';
9547: { Simulation of a probability density function by Metropolis-Hastings }
9548: procedure InitGAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9549: { Initializes Simulated Annealing parameters }
9550: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9551: { Initializes log file }
9552: procedure SimAnn(Func      : TFuncNVar;
9553:                  X, Xmin, Xmax : TVector;
9554:                  Lb, Ub      : Integer;
9555:                  var F_min    : Float); external 'dmath';
9556: { Minimization of a function of several var. by simulated annealing }
9557: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9558: { Initializes Genetic Algorithm parameters }
9559: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9560: { Initializes log file }
9561: procedure GenAlg(Func      : TFuncNVar;
9562:                  X, Xmin, Xmax : TVector;
9563:                  Lb, Ub      : Integer;
9564:                  var F_min    : Float); external 'dmath';
9565: { Minimization of a function of several var. by genetic algorithm }
9566: { -----
9567:   Statistics
9568:   ----- }
9569: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9570: { Mean of sample X }
9571: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9572: { Minimum of sample X }
9573: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9574: { Maximum of sample X }
9575: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9576: { Median of sample X }
9577: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9578: { Standard deviation estimated from sample X }
9579: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9580: { Standard deviation of population }
9581: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9582: { Correlation coefficient }
9583: function Skewness(X : TVector; Lb, Ub : Integer; M, Sigma: Float) : Float; external 'dmath';
9584: { Skewness of sample X }
9585: function Kurtosis(X : TVector; Lb, Ub : Integer; M, Sigma: Float) : Float; external 'dmath';
9586: { Kurtosis of sample X }
9587: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9588: { Quick sort (ascending order) }
9589: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9590: { Quick sort (descending order) }
9591: procedure Interval(X1, X2      : Float;
9592:                   MinDiv, MaxDiv : Integer;
9593:                   var Min, Max, Step : Float); external 'dmath';
9594: { Determines an interval for a set of values }
9595: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9596:                   var XMin, XMax, XStep : Float); external 'dmath';
9597: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9598: procedure StudIndep(N1, N2      : Integer;
9599:                   M1, M2, S1, S2 : Float;
9600:                   var T         : Float;
9601:                   var DoF       : Integer); external 'dmath';
9602: { Student t-test for independent samples }
9603: procedure StudPaired(X, Y      : TVector;
9604:                   Lb, Ub      : Integer;
9605:                   var T       : Float;
9606:                   var DoF     : Integer); external 'dmath';
9607: { Student t-test for paired samples }
9608: procedure AnOVal(Ns      : Integer;
9609:                 N         : TIntVector;
9610:                 M, S      : TVector;
9611:                 var V_f, V_r, F : Float);

```

```

9612:         var DoF_f, DoF_r : Integer); external 'dmath';
9613: { One-way analysis of variance }
9614: procedure AnOVA2(NA, NB, Nobs : Integer;
9615:                 M, S       : TMatrix;
9616:                 V, F       : TVector;
9617:                 DoF        : TIntVector); external 'dmath';
9618: { Two-way analysis of variance }
9619: procedure Snedecor(N1, N2      : Integer;
9620:                  S1, S2      : Float;
9621:                  var F       : Float;
9622:                  var DoF1, DoF2 : Integer); external 'dmath';
9623: { Snedecor's F-test (comparison of two variances) }
9624: procedure Bartlett(Ns      : Integer;
9625:                   N       : TIntVector;
9626:                   S       : TVector;
9627:                   var Khi2 : Float;
9628:                   var DoF  : Integer); external 'dmath';
9629: { Bartlett's test (comparison of several variances) }
9630: procedure Mann_Whitney(N1, N2 : Integer;
9631:                      X1, X2  : TVector;
9632:                      var U, Eps : Float); external 'dmath';
9633: { Mann-Whitney test }
9634: procedure Wilcoxon(X, Y      : TVector;
9635:                  Lb, Ub     : Integer;
9636:                  var Ndiff  : Integer;
9637:                  var T, Eps : Float); external 'dmath';
9638: { Wilcoxon test }
9639: procedure Kruskal_Wallis(Ns      : Integer;
9640:                        N       : TIntVector;
9641:                        X       : TMatrix;
9642:                        var H    : Float;
9643:                        var DoF  : Integer); external 'dmath';
9644: { Kruskal-Wallis test }
9645: procedure Khi2_Conform(N_cls : Integer;
9646:                      N_estim : Integer;
9647:                      Obs     : TIntVector;
9648:                      Calc    : TVector;
9649:                      var Khi2 : Float;
9650:                      var DoF  : Integer); external 'dmath';
9651: { Khi-2 test for conformity }
9652: procedure Khi2_Indep(N_lin : Integer;
9653:                    N_col  : Integer;
9654:                    Obs    : TIntMatrix;
9655:                    var Khi2 : Float;
9656:                    var DoF  : Integer); external 'dmath';
9657: { Khi-2 test for independence }
9658: procedure Woolf_Conform(N_cls : Integer;
9659:                      N_estim : Integer;
9660:                      Obs     : TIntVector;
9661:                      Calc    : TVector;
9662:                      var G    : Float;
9663:                      var DoF  : Integer); external 'dmath';
9664: { Woolf's test for conformity }
9665: procedure Woolf_Indep(N_lin : Integer;
9666:                    N_col  : Integer;
9667:                    Obs    : TIntMatrix;
9668:                    var G    : Float;
9669:                    var DoF  : Integer); external 'dmath';
9670: { Woolf's test for independence }
9671: procedure DimStatClassVector(var C : TStatClassVector;
9672:                             Ub   : Integer); external 'dmath';
9673: { Allocates an array of statistical classes: C[0..Ub] }
9674: procedure Distrib(X      : TVector;
9675:                 Lb, Ub  : Integer;
9676:                 A, B, H : Float;
9677:                 C       : TStatClassVector); external 'dmath';
9678: { Distributes an array X[Lb..Ub] into statistical classes }
9679: { -----
9680:   Linear / polynomial regression
9681:   ----- }
9682: procedure LinFit(X, Y      : TVector;
9683:                Lb, Ub     : Integer;
9684:                B          : TVector;
9685:                V          : TMatrix); external 'dmath';
9686: { Linear regression : Y = B(0) + B(1) * X }
9687: procedure WLinFit(X, Y, S : TVector;
9688:                 Lb, Ub  : Integer;
9689:                 B       : TVector;
9690:                 V       : TMatrix); external 'dmath';
9691: { Weighted linear regression : Y = B(0) + B(1) * X }
9692: procedure SVDLinFit(X, Y : TVector;
9693:                   Lb, Ub : Integer;
9694:                   SVDTol : Float;
9695:                   B      : TVector;
9696:                   V      : TMatrix); external 'dmath';
9697: { Unweighted linear regression by singular value decomposition }
9698: procedure WSVDLinFit(X, Y, S : TVector;
9699:                   Lb, Ub  : Integer;
9700:                   SVDTol : Float;

```



```

9701:         B      : TVector;
9702:         V      : TMatrix); external 'dmath';
9703: { Weighted linear regression by singular value decomposition }
9704: procedure MulFit(X      : TMatrix;
9705:                 Y      : TVector;
9706:                 Lb, Ub, Nvar : Integer;
9707:                 ConstTerm : Boolean;
9708:                 B      : TVector;
9709:                 V      : TMatrix); external 'dmath';
9710: { Multiple linear regression by Gauss-Jordan method }
9711: procedure WMulFit(X      : TMatrix;
9712:                  Y, S    : TVector;
9713:                  Lb, Ub, Nvar : Integer;
9714:                  ConstTerm : Boolean;
9715:                  B      : TVector;
9716:                  V      : TMatrix); external 'dmath';
9717: { Weighted multiple linear regression by Gauss-Jordan method }
9718: procedure SVDFit(X      : TMatrix;
9719:                  Y      : TVector;
9720:                  Lb, Ub, Nvar : Integer;
9721:                  ConstTerm : Boolean;
9722:                  SVDTol   : Float;
9723:                  B      : TVector;
9724:                  V      : TMatrix); external 'dmath';
9725: { Multiple linear regression by singular value decomposition }
9726: procedure WSVDFit(X      : TMatrix;
9727:                   Y, S    : TVector;
9728:                   Lb, Ub, Nvar : Integer;
9729:                   ConstTerm : Boolean;
9730:                   SVDTol   : Float;
9731:                   B      : TVector;
9732:                   V      : TMatrix); external 'dmath';
9733: { Weighted multiple linear regression by singular value decomposition }
9734: procedure PolFit(X, Y      : TVector;
9735:                  Lb, Ub, Deg : Integer;
9736:                  B      : TVector;
9737:                  V      : TMatrix); external 'dmath';
9738: { Polynomial regression by Gauss-Jordan method }
9739: procedure WPolFit(X, Y, S  : TVector;
9740:                  Lb, Ub, Deg : Integer;
9741:                  B      : TVector;
9742:                  V      : TMatrix); external 'dmath';
9743: { Weighted polynomial regression by Gauss-Jordan method }
9744: procedure SVDPolFit(X, Y      : TVector;
9745:                    Lb, Ub, Deg : Integer;
9746:                    SVDTol   : Float;
9747:                    B      : TVector;
9748:                    V      : TMatrix); external 'dmath';
9749: { Unweighted polynomial regression by singular value decomposition }
9750: procedure WSVDPolFit(X, Y, S  : TVector;
9751:                     Lb, Ub, Deg : Integer;
9752:                     SVDTol   : Float;
9753:                     B      : TVector;
9754:                     V      : TMatrix); external 'dmath';
9755: { Weighted polynomial regression by singular value decomposition }
9756: procedure RegTest(Y, Ycalc : TVector;
9757:                  LbY, UbY : Integer;
9758:                  V      : TMatrix;
9759:                  LbV, UbV : Integer;
9760:                  var Test : TRegTest); external 'dmath';
9761: { Test of unweighted regression }
9762: procedure WRegTest(Y, Ycalc, S : TVector;
9763:                   LbY, UbY    : Integer;
9764:                   V          : TMatrix;
9765:                   LbV, UbV    : Integer;
9766:                   var Test    : TRegTest); external 'dmath';
9767: { Test of weighted regression }
9768: { -----
9769: Nonlinear regression
9770: ----- }
9771: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9772: { Sets the optimization algorithm for nonlinear regression }
9773: function GetOptAlgo : TOptAlgo; external 'dmath';
9774: { Returns the optimization algorithm }
9775: procedure SetMaxParam(N : Byte); external 'dmath';
9776: { Sets the maximum number of regression parameters for nonlinear regression }
9777: function GetMaxParam : Byte; external 'dmath';
9778: { Returns the maximum number of regression parameters for nonlinear regression }
9779: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9780: { Sets the bounds on the I-th regression parameter }
9781: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
9782: { Returns the bounds on the I-th regression parameter }
9783: procedure NLFit(RegFunc : TRegFunc;
9784:                DerivProc : TDerivProc;
9785:                X, Y      : TVector;
9786:                Lb, Ub    : Integer;
9787:                MaxIter   : Integer;
9788:                Tol       : Float;
9789:                B          : TVector;

```

```

9790:         FirstPar,
9791:         LastPar   : Integer;
9792:         V         : TMatrix); external 'dmath';
9793: { Unweighted nonlinear regression }
9794: procedure WNLFit(RegFunc : TRegFunc;
9795:                 DerivProc : TDerivProc;
9796:                 X, Y, S   : TVector;
9797:                 Lb, Ub    : Integer;
9798:                 MaxIter   : Integer;
9799:                 Tol       : Float;
9800:                 B         : TVector;
9801:                 FirstPar,
9802:                 LastPar   : Integer;
9803:                 V         : TMatrix); external 'dmath';
9804: { Weighted nonlinear regression }
9805: procedure SetMCFile(FileName : String); external 'dmath';
9806: { Set file for saving MCMC simulations }
9807: procedure SimFit(RegFunc : TRegFunc;
9808:                 X, Y     : TVector;
9809:                 Lb, Ub    : Integer;
9810:                 B         : TVector;
9811:                 FirstPar,
9812:                 LastPar   : Integer;
9813:                 V         : TMatrix); external 'dmath';
9814: { Simulation of unweighted nonlinear regression by MCMC }
9815: procedure WSimFit(RegFunc : TRegFunc;
9816:                 X, Y, S   : TVector;
9817:                 Lb, Ub    : Integer;
9818:                 B         : TVector;
9819:                 FirstPar,
9820:                 LastPar   : Integer;
9821:                 V         : TMatrix); external 'dmath';
9822: { Simulation of weighted nonlinear regression by MCMC }
9823: { -----
9824:   Nonlinear regression models
9825: ----- }
9826: procedure FracFit(X, Y : TVector;
9827:                 Lb, Ub : Integer;
9828:                 Deg1, Deg2 : Integer;
9829:                 ConsTerm : Boolean;
9830:                 MaxIter  : Integer;
9831:                 Tol      : Float;
9832:                 B        : TVector;
9833:                 V        : TMatrix); external 'dmath';
9834: { Unweighted fit of rational fraction }
9835: procedure WFracFit(X, Y, S : TVector;
9836:                 Lb, Ub : Integer;
9837:                 Deg1, Deg2 : Integer;
9838:                 ConsTerm : Boolean;
9839:                 MaxIter  : Integer;
9840:                 Tol      : Float;
9841:                 B        : TVector;
9842:                 V        : TMatrix); external 'dmath';
9843: { Weighted fit of rational fraction }
9844:
9845: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9846: { Returns the value of the rational fraction at point X }
9847: procedure ExpFit(X, Y : TVector;
9848:                 Lb, Ub, Nexp : Integer;
9849:                 ConsTerm    : Boolean;
9850:                 MaxIter     : Integer;
9851:                 Tol         : Float;
9852:                 B           : TVector;
9853:                 V           : TMatrix); external 'dmath';
9854: { Unweighted fit of sum of exponentials }
9855: procedure WExpFit(X, Y, S : TVector;
9856:                 Lb, Ub, Nexp : Integer;
9857:                 ConsTerm    : Boolean;
9858:                 MaxIter     : Integer;
9859:                 Tol         : Float;
9860:                 B           : TVector;
9861:                 V           : TMatrix); external 'dmath';
9862: { Weighted fit of sum of exponentials }
9863: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9864: { Returns the value of the regression function at point X }
9865: procedure IncExpFit(X, Y : TVector;
9866:                 Lb, Ub : Integer;
9867:                 ConsTerm : Boolean;
9868:                 MaxIter  : Integer;
9869:                 Tol      : Float;
9870:                 B        : TVector;
9871:                 V        : TMatrix); external 'dmath';
9872: { Unweighted fit of model of increasing exponential }
9873: procedure WIncExpFit(X, Y, S : TVector;
9874:                 Lb, Ub : Integer;
9875:                 ConsTerm : Boolean;
9876:                 MaxIter  : Integer;
9877:                 Tol      : Float;
9878:                 B        : TVector;

```

```

9879:         V      : TMatrix); external 'dmath';
9880: { Weighted fit of increasing exponential }
9881: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9882: { Returns the value of the regression function at point X }
9883: procedure ExpLinFit(X, Y      : TVector;
9884:         Lb, Ub      : Integer;
9885:         MaxIter      : Integer;
9886:         Tol          : Float;
9887:         B            : TVector;
9888:         V            : TMatrix); external 'dmath';
9889: { Unweighted fit of the "exponential + linear" model }
9890: procedure WExpLinFit(X, Y, S : TVector;
9891:         Lb, Ub      : Integer;
9892:         MaxIter      : Integer;
9893:         Tol          : Float;
9894:         B            : TVector;
9895:         V            : TMatrix); external 'dmath';
9896: { Weighted fit of the "exponential + linear" model }
9897:
9898: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9899: { Returns the value of the regression function at point X }
9900: procedure MichFit(X, Y      : TVector;
9901:         Lb, Ub      : Integer;
9902:         MaxIter      : Integer;
9903:         Tol          : Float;
9904:         B            : TVector;
9905:         V            : TMatrix); external 'dmath';
9906: { Unweighted fit of Michaelis equation }
9907: procedure WMichFit(X, Y, S : TVector;
9908:         Lb, Ub      : Integer;
9909:         MaxIter      : Integer;
9910:         Tol          : Float;
9911:         B            : TVector;
9912:         V            : TMatrix); external 'dmath';
9913: { Weighted fit of Michaelis equation }
9914: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9915: { Returns the value of the Michaelis equation at point X }
9916: procedure MintFit(X, Y      : TVector;
9917:         Lb, Ub      : Integer;
9918:         MintVar      : TMintVar;
9919:         Fit_S0       : Boolean;
9920:         MaxIter      : Integer;
9921:         Tol          : Float;
9922:         B            : TVector;
9923:         V            : TMatrix); external 'dmath';
9924: { Unweighted fit of the integrated Michaelis equation }
9925: procedure WMintFit(X, Y, S : TVector;
9926:         Lb, Ub      : Integer;
9927:         MintVar      : TMintVar;
9928:         Fit_S0       : Boolean;
9929:         MaxIter      : Integer;
9930:         Tol          : Float;
9931:         B            : TVector;
9932:         V            : TMatrix); external 'dmath';
9933: { Weighted fit of the integrated Michaelis equation }
9934: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9935: { Returns the value of the integrated Michaelis equation at point X }
9936: procedure HillFit(X, Y      : TVector;
9937:         Lb, Ub      : Integer;
9938:         MaxIter      : Integer;
9939:         Tol          : Float;
9940:         B            : TVector;
9941:         V            : TMatrix); external 'dmath';
9942: { Unweighted fit of Hill equation }
9943: procedure WHillFit(X, Y, S : TVector;
9944:         Lb, Ub      : Integer;
9945:         MaxIter      : Integer;
9946:         Tol          : Float;
9947:         B            : TVector;
9948:         V            : TMatrix); external 'dmath';
9949: { Weighted fit of Hill equation }
9950: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9951: { Returns the value of the Hill equation at point X }
9952: procedure LogiFit(X, Y      : TVector;
9953:         Lb, Ub      : Integer;
9954:         ConsTerm     : Boolean;
9955:         General       : Boolean;
9956:         MaxIter      : Integer;
9957:         Tol          : Float;
9958:         B            : TVector;
9959:         V            : TMatrix); external 'dmath';
9960: { Unweighted fit of logistic function }
9961: procedure WLogiFit(X, Y, S : TVector;
9962:         Lb, Ub      : Integer;
9963:         ConsTerm     : Boolean;
9964:         General       : Boolean;
9965:         MaxIter      : Integer;
9966:         Tol          : Float;
9967:         B            : TVector;

```

```

9968:          V          : TMatrix); external 'dmath';
9969: { Weighted fit of logistic function }
9970: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9971: { Returns the value of the logistic function at point X }
9972: procedure PKFit(X, Y      : TVector;
9973:               Lb, Ub    : Integer;
9974:               MaxIter   : Integer;
9975:               Tol       : Float;
9976:               B         : TVector;
9977:               V         : TMatrix); external 'dmath';
9978: { Unweighted fit of the acid-base titration curve }
9979: procedure WPKFit(X, Y, S : TVector;
9980:               Lb, Ub    : Integer;
9981:               MaxIter   : Integer;
9982:               Tol       : Float;
9983:               B         : TVector;
9984:               V         : TMatrix); external 'dmath';
9985: { Weighted fit of the acid-base titration curve }
9986: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9987: { Returns the value of the acid-base titration function at point X }
9988: procedure PowFit(X, Y      : TVector;
9989:               Lb, Ub    : Integer;
9990:               MaxIter   : Integer;
9991:               Tol       : Float;
9992:               B         : TVector;
9993:               V         : TMatrix); external 'dmath';
9994: { Unweighted fit of power function }
9995: procedure WPowFit(X, Y, S : TVector;
9996:               Lb, Ub    : Integer;
9997:               MaxIter   : Integer;
9998:               Tol       : Float;
9999:               B         : TVector;
10000:              V         : TMatrix); external 'dmath';
10001: { Weighted fit of power function }
10002:
10003: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10004: { Returns the value of the power function at point X }
10005: procedure GammaFit(X, Y      : TVector;
10006:               Lb, Ub    : Integer;
10007:               MaxIter   : Integer;
10008:               Tol       : Float;
10009:               B         : TVector;
10010:               V         : TMatrix); external 'dmath';
10011: { Unweighted fit of gamma distribution function }
10012: procedure WGammaFit(X, Y, S : TVector;
10013:               Lb, Ub    : Integer;
10014:               MaxIter   : Integer;
10015:               Tol       : Float;
10016:               B         : TVector;
10017:               V         : TMatrix); external 'dmath';
10018: { Weighted fit of gamma distribution function }
10019: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10020: { Returns the value of the gamma distribution function at point X }
10021: { -----
10022:   Principal component analysis
10023:   ----- }
10024: procedure VecMean(X          : TMatrix;
10025:               Lb, Ub, Nvar : Integer;
10026:               M          : TVector); external 'dmath';
10027: { Computes the mean vector M from matrix X }
10028: procedure VecSD(X          : TMatrix;
10029:               Lb, Ub, Nvar : Integer;
10030:               M, S        : TVector); external 'dmath';
10031: { Computes the vector of standard deviations S from matrix X }
10032: procedure MatVarCov(X          : TMatrix;
10033:               Lb, Ub, Nvar : Integer;
10034:               M          : TVector;
10035:               V          : TMatrix); external 'dmath';
10036: { Computes the variance-covariance matrix V from matrix X }
10037: procedure MatCorrel(V          : TMatrix;
10038:               Nvar : Integer;
10039:               R          : TMatrix); external 'dmath';
10040: { Computes the correlation matrix R from the var-cov matrix V }
10041: procedure PCA(R          : TMatrix;
10042:               Nvar      : Integer;
10043:               Lambda    : TVector;
10044:               C, Rc     : TMatrix); external 'dmath';
10045: { Performs a principal component analysis of the correlation matrix R }
10046: procedure ScaleVar(X          : TMatrix;
10047:               Lb, Ub, Nvar : Integer;
10048:               M, S        : TVector;
10049:               Z          : TMatrix); external 'dmath';
10050: { Scales a set of variables by subtracting means and dividing by SD's }
10051: procedure PrinFac(Z          : TMatrix;
10052:               Lb, Ub, Nvar : Integer;
10053:               C, F        : TMatrix); external 'dmath';
10054: { Computes principal factors }
10055: { -----
10056:   Strings

```



```

10057: ----- }
10058: function LTrim(S : String) : String; external 'dmath';
10059: { Removes leading blanks }
10060: function RTrim(S : String) : String; external 'dmath';
10061: { Removes trailing blanks }
10062: function Trim(S : String) : String; external 'dmath';
10063: { Removes leading and trailing blanks }
10064: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10065: { Returns a string made of character C repeated N times }
10066: function RFill(S : String; L : Byte) : String; external 'dmath';
10067: { Completes string S with trailing blanks for a total length L }
10068: function LFill(S : String; L : Byte) : String; external 'dmath';
10069: { Completes string S with leading blanks for a total length L }
10070: function CFill(S : String; L : Byte) : String; external 'dmath';
10071: { Centers string S on a total length L }
10072: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10073: { Replaces in string S all the occurrences of C1 by C2 }
10074: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10075: { Extracts a field from a string }
10076: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10077: { Parses a string into its constitutive fields }
10078: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10079: { Sets the numeric format }
10080: function FloatStr(X : Float) : String; external 'dmath';
10081: { Converts a real to a string according to the numeric format }
10082: function IntStr(N : LongInt) : String; external 'dmath';
10083: { Converts an integer to a string }
10084: function CompStr(Z : Complex) : String; external 'dmath';
10085: { Converts a complex number to a string }
10086: {$IFDEF DELPHI}
10087: function StrDec(S : String) : String; external 'dmath';
10088: { Set decimal separator to the symbol defined in SysUtils }
10089: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10090: { Test if a string represents a number and returns it in X }
10091: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10092: { Reads a floating point number from an Edit control }
10093: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10094: { Writes a floating point number in a text file }
10095: {$ENDIF}
10096: {
10097:   BGI / Delphi graphics
10098: }
10099: function InitGraphics
10100: {$IFDEF DELPHI}
10101: (Width, Height : Integer) : Boolean;
10102: {$ELSE}
10103: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10104: { Enters graphic mode }
10105: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10106:   X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10107: { Sets the graphic window }
10108: procedure SetOxScale(Scale : TScale;
10109:   OxMin, OxMax, OxStep : Float); external 'dmath';
10110: { Sets the scale on the Ox axis }
10111: procedure SetOyScale(Scale : TScale;
10112:   OyMin, OyMax, OyStep : Float); external 'dmath';
10113: { Sets the scale on the Oy axis }
10114: procedure GetOxScale(var Scale : TScale;
10115:   var OxMin, OxMax, OxStep : Float); external 'dmath';
10116: { Returns the scale on the Ox axis }
10117: procedure GetOyScale(var Scale : TScale;
10118:   var OyMin, OyMax, OyStep : Float); external 'dmath';
10119: { Returns the scale on the Oy axis }
10120: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10121: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10122: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10123: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10124: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10125: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10126: {$IFDEF DELPHI}
10127: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10128: { Sets the font for the main graph title }
10129: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10130: { Sets the font for the Ox axis (title and labels) }
10131: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10132: { Sets the font for the Oy axis (title and labels) }
10133: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10134: { Sets the font for the legends }
10135: procedure SetClipping(Clip : Boolean); external 'dmath';
10136: { Determines whether drawings are clipped at the current viewport
10137:   boundaries, according to the value of the Boolean parameter Clip }
10138: {$ENDIF}
10139: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF} external 'dmath';
10140: { Plots the horizontal axis }
10141: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF} external 'dmath';
10142: { Plots the vertical axis }
10143: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10144: { Plots a grid on the graph }
10145: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF} external 'dmath';

```

```

10146: { Writes the title of the graph }
10147: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10148: { Sets the maximum number of curves and re-initializes their parameters }
10149: procedure SetPointParam
10150: {$IFDEF DELPHI}
10151: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10152: {$ELSE}
10153: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10154: { Sets the point parameters for curve # CurvIndex }
10155: procedure SetLineParam
10156: {$IFDEF DELPHI}
10157: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10158: {$ELSE}
10159: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10160: { Sets the line parameters for curve # CurvIndex }
10161: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10162: { Sets the legend for curve # CurvIndex }
10163: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10164: { Sets the step for curve # CurvIndex }
10165: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10166: procedure GetPointParam
10167: {$IFDEF DELPHI}
10168: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10169: {$ELSE}
10170: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10171: { Returns the point parameters for curve # CurvIndex }
10172: procedure GetLineParam
10173: {$IFDEF DELPHI}
10174: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10175: {$ELSE}
10176: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10177: { Returns the line parameters for curve # CurvIndex }
10178: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10179: { Returns the legend for curve # CurvIndex }
10180: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10181: { Returns the step for curve # CurvIndex }
10182: {$IFDEF DELPHI}
10183: procedure PlotPoint(Canvas : TCanvas;
10184:                     X, Y : Float; CurvIndex : Integer); external 'dmath';
10185: {$ELSE}
10186: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10187: {$ENDIF}
10188: { Plots a point on the screen }
10189: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10190:                    X, Y : TVector;
10191:                    Lb, Ub, CurvIndex : Integer); external 'dmath';
10192: { Plots a curve }
10193: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10194:                                  X, Y, S : TVector;
10195:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10196: { Plots a curve with error bars }
10197: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10198:                   Func : TFunc;
10199:                   Xmin, Xmax : Float;
10200:                   {$IFDEF DELPHI}Npt : Integer;{$ENDIF}
10201:                   CurvIndex : Integer); external 'dmath';
10202: { Plots a function }
10203: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10204:                      NCurv : Integer;
10205:                      ShowPoints, ShowLines : Boolean); external 'dmath';
10206: { Writes the legends for the plotted curves }
10207: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10208:                 Nx, Ny, Nc : Integer;
10209:                 X, Y, Z : TVector;
10210:                 F : TMatrix); external 'dmath';
10211: { Contour plot }
10212: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10213: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10214: function Xuser(X : Integer):Float; external 'dmath'; {Converts screen coordinate X to user abscissa }
10215: function Yuser(Y : Integer):Float; external 'dmath'; {Converts screen coordinate Y to user ordinate }
10216: {$IFDEF DELPHI}
10217: procedure LeaveGraphics; external 'dmath';
10218: { Quits graphic mode }
10219: {$ENDIF}
10220: { -----
10221: LaTeX graphics
10222: ----- }
10223: function TeX_InitGraphics(FileName : String; PgWidth, PgHeight : Integer;
10224:                           Header : Boolean) : Boolean; external 'dmath';
10225: { Initializes the LaTeX file }
10226: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10227: { Sets the graphic window }
10228: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10229: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10230: { Sets the scale on the Ox axis }
10231: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10232: { Sets the scale on the Oy axis }
10233: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10234: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }

```

```

10235: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10236: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10237: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10238: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10239: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10240: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10241: { Sets the maximum number of curves and re-initializes their parameters }
10242: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10243: { Sets the point parameters for curve # CurvIndex }
10244: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10245:                             Width : Float; Smooth : Boolean); external 'dmath';
10246: { Sets the line parameters for curve # CurvIndex }
10247: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10248: { Sets the legend for curve # CurvIndex }
10249: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10250: { Sets the step for curve # CurvIndex }
10251: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10252: { Plots a curve }
10253: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10254:                                       Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10255: { Plots a curve with error bars }
10256: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10257:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
10258: { Plots a function }
10259: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10260: { Writes the legends for the plotted curves }
10261: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix); external 'dmath';
10262: { Contour plot }
10263: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10264: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10265:
10266: //*****unit uPSI_SynPdf;
10267: function RawUTF8ToPDFString( const Value : RawUTF8 ) : PDFString
10268: function _DateToPdfDate( ADate : TDateTime ) : TPdfDate
10269: function _PdfDateToDateTime( const AText : TPdfDate ) : TDateTime
10270: function PdfRect( Left, Top, Right, Bottom : Single ) : TPdfRect;
10271: function PdfRect1( const Box : TPdfBox ) : TPdfRect;
10272: function PdfBox( Left, Top, Width, Height : Single ) : TPdfBox
10273: //Function _GetCharCount( Text : PAnsiChar ) : integer
10274: //Procedure L2R( W : PWideChar; L : integer )
10275: function PdfCoord( MM : single ) : integer
10276: function CurrentPrinterPaperSize : TPDFPaperSize
10277: function CurrentPrinterRes : TPoint
10278: procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8 )
10279: procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer )
10280: procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect )
10281: Const('Usp10','String').SetString( 'usp10.dll
10282: AddTypes('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10283: 'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10284: AddTypes('TScriptState_set', 'set of TScriptState_enum
10285: //*****
10286:
10287: procedure SIRegister_PMrand(CL: TPSPascalCompiler); //ParkMiller
10288: begin
10289:   Procedure PMrandomize( I : word)
10290:   Function PMrandom : longint
10291:   Function Rrand : extended
10292:   Function Irand( N : word) : word
10293:   Function Brand( P : extended) : boolean
10294:   Function Nrand : extended
10295: end;
10296:
10297: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10298: begin
10299:   Function Endian( x : LongWord) : LongWord
10300:   Function Endian64( x : Int64) : Int64
10301:   Function spRol( x : LongWord; y : Byte) : LongWord
10302:   Function spRor( x : LongWord; y : Byte) : LongWord
10303:   Function Ror64( x : Int64; y : Byte) : Int64
10304: end;
10305:
10306: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10307: begin
10308:   Procedure ClearModules
10309:   Procedure ReadMapFile( Fname : string)
10310:   Function AddressInfo( Address : dword) : string
10311: end;
10312:
10313: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10314: begin
10315:   TTarPermission, '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10316:   + 'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10317:   + 'teByOther, tpExecuteByOther )
10318:   TTarPermissions, 'set of TTarPermission
10319:   TFileType, '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10320:   + 'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10321:   TTarMode, '( tmSetUid, tmSetGid, tmSaveText )
10322:   TTarModes, 'set of TTarMode
10323:   TTarDirRec, 'record Name : STRING; Size : INT64; DateTime : TDA'

```

```

10324:  + 'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10325:  + 'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10326:  + 'ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10327:  + 'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10328:  SIRegister_TTarArchive(CL);
10329:  SIRegister_TTarWriter(CL);
10330:  Function PermissionString( Permissions : TTarPermissions) : STRING
10331:  Function ConvertFilename( Filename : STRING) : STRING
10332:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10333:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10334:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10335: end;
10336:
10337:
10338: //*****unit uPSI_TlHelp32;
10339: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10340: begin
10341:   Const('MAX_MODULE_NAME32','LongInt').SetInt( 255);
10342:   Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10343:   Const('TH32CS_SNAPHEAPLIST','LongWord').SetUInt( $00000001);
10344:   Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10345:   Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10346:   Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10347:   Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10348:   tagHEAPLIST32, record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end;
10349:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10350:   AddTypeS('TheapList32', 'tagHEAPLIST32
10351:   Const('HF32_DEFAULT','LongInt').SetInt( 1);
10352:   Const('HF32_SHARED','LongInt').SetInt( 2);
10353:   Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10354:   Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10355:   AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10356:   + 'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10357:   + 'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10358:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10359:   AddTypeS('TheapEntry32', 'tagHEAPENTRY32
10360:   Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10361:   Const('LF32_FREE','LongWord').SetUInt( $00000002);
10362:   Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10363:   Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10364:   Function Heap32Next( var lphe : THeapEntry32) : BOOL
10365:   DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10366:   AddTypeS('tagTHREADEENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10367:   + '2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10368:   + 'aPri : Longint; dwFlags : DWORD; end
10369:   AddTypeS('THREADEENTRY32', 'tagTHREADEENTRY32
10370:   AddTypeS('TThreadEntry32', 'tagTHREADEENTRY32
10371:   Function Thread32First( hSnapshot : THandle; var lpthe : TThreadEntry32) : BOOL
10372:   Function Thread32Next( hSnapshot : THandle; var lpthe : TThreadEntry32) : BOOL
10373: end;
10374:   Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10375:   Const('EW_REBOOTSYSTEM','LongWord').SetUInt( $0043);
10376:   Const('EW_EXITANDEXECAPP','LongWord').SetUInt( $0044);
10377:   Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10378:   Const('EWX_LOGOFF','LongInt').SetInt( 0);
10379:   Const('EWX_SHUTDOWN','LongInt').SetInt( 1);
10380:   Const('EWX_REBOOT','LongInt').SetInt( 2);
10381:   Const('EWX_FORCE','LongInt').SetInt( 4);
10382:   Const('EWX_POWEROFF','LongInt').SetInt( 8);
10383:   Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10384:   Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10385:   Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10386:   Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10387:   Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10388:   Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10389:   Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10390:   Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10391:   Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10392:   Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10393:   Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10394:   Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10395:   Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10396:   Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10397:   Function GetDesktopWindow : HWND
10398:   Function GetParent( hWnd : HWND) : HWND
10399:   Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10400:   Function GetTopWindow( hWnd : HWND) : HWND
10401:   Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10402:   Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10403: //Delphi DFM
10404: Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10405: Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10406: procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10407: function GetHighlightersFilter(AHighlighters: TStringList) : string;
10408: function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10409: Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10410: Function OpenIcon( hWnd : HWND) : BOOL
10411: Function CloseWindow( hWnd : HWND) : BOOL
10412: Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL

```



```

10413: Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10414: Function IsWindowVisible( hWnd : HWND) : BOOL
10415: Function IsIconic( hWnd : HWND) : BOOL
10416: Function AnyPopup : BOOL
10417: Function BringWindowToTop( hWnd : HWND) : BOOL
10418: Function IsZoomed( hWnd : HWND) : BOOL
10419: Function IsWindow( hWnd : HWND) : BOOL
10420: Function IsMenu( hMenu : HMENU) : BOOL
10421: Function IsChild( hWndParent, hWnd : HWND) : BOOL
10422: Function DestroyWindow( hWnd : HWND) : BOOL
10423: Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10424: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10425: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10426: Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10427: Function IsWindowUnicode( hWnd : HWND) : BOOL
10428: Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10429: Function IsWindowEnabled( hWnd : HWND) : BOOL
10430:
10431: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10432: begin
10433:   const ('ShowSetupDialogOptLong','String').SetString( '--setup
10434:   PrimaryConfPathOptLong','String').SetString( '--primary-config-path=
10435:   PrimaryConfPathOptShort','String').SetString( '--pcp=
10436:   SecondaryConfPathOptLong','String').SetString( '--secondary-config-path=
10437:   SecondaryConfPathOptShort','String').SetString( '--scp=
10438:   NoSplashScreenOptLong','String').SetString( '--no-splash-screen
10439:   NoSplashScreenOptShort','String').SetString( '--nsc
10440:   StartedByStartLazarusOpt','String').SetString( '--started-by-startlazarus
10441:   SkipLastProjectOpt','String').SetString( '--skip-last-project
10442:   DebugLogOpt','String').SetString( '--debug-log=
10443:   DebugLogOptEnable','String').SetString( '--debug-enable=
10444:   LanguageOpt','String').SetString( '--language=
10445:   LazarusDirOpt','String').SetString( '--lazarusdir=
10446: Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10447: Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10448: Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10449: Function IsHelpRequested : Boolean
10450: Function IsVersionRequested : boolean
10451: Function GetLanguageSpecified : string
10452: Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10453: Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10454: Procedure ParseNoGuiCmdLineParams
10455: Function ExtractCmdLineFileNames : TStrings
10456: end;
10457:
10458:
10459: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10460: begin
10461:   Function CompareFileNames( const Filename1, Filename2 : string) : integer
10462:   Function CompareFileNamesIgnoreCase( const Filename1, Filename2 : string) : integer
10463:   Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10464:   Function CompareFileExt1( const Filename, Ext : string) : integer;
10465:   Function CompareFileNamesStarts( const Filename1, Filename2 : string) : integer
10466:   Function CompareFileNames(Filename:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10467:   Function CompareFileNamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10468:   Function DirPathExists( DirectoryName : string) : boolean
10469:   Function DirectoryIsWritable( const DirectoryName : string) : boolean
10470:   Function ExtractFileNameOnly( const AFilename : string) : string
10471:   Function FilenameIsAbsolute( const TheFilename : string) : boolean
10472:   Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10473:   Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10474:   Function ForceDirectory( DirectoryName : string) : boolean
10475: Procedure CheckIfFileIsExecutable( const AFilename : string)
10476: Procedure CheckIfFileIsSymlink( const AFilename : string)
10477: Function FileIsText( const AFilename : string) : boolean
10478: Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10479: Function FilenameIsTrimmed( const TheFilename : string) : boolean
10480: Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10481: Function TrimFilename( const AFilename : string) : string
10482: Function ResolveDots( const AFilename : string) : string
10483: Procedure ForcePathDelims( var FileName : string)
10484: Function GetForcedPathDelims( const FileName : string) : String
10485: Function CleanAndExpandFilename( const Filename : string) : string
10486: Function CleanAndExpandDirectory( const Filename : string) : string
10487: Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10488: Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10489: Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10490: Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
AlwaysRequireSharedBaseFolder: Boolean) : string
10491: Function FileIsInPath( const Filename, Path : string) : boolean
10492: Function AppendPathDelim( const Path : string) : string
10493: Function ChompPathDelim( const Path : string) : string
10494: Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10495: Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10496: Function MinimizeSearchPath( const SearchPath : string) : string
10497: Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10498: (*Function FileExistsUTF8( const Filename : string) : boolean
10499: Function FileAgeUTF8( const FileName : string) : Longint

```

```

10500: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
10501: Function ExpandFileNameUTF8( const FileName : string; BaseDir : string ) : string
10502: Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10503: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10504: Procedure FindCloseUTF8( var F : TSearchRec)
10505: Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10506: Function FileGetAttrUTF8( const FileName : String) : Longint
10507: Function FileSetAttrUTF8( const FileName : String; Attr : longint) : Longint
10508: Function DeleteFileUTF8( const FileName : String) : Boolean
10509: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10510: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10511: Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10512: Function GetCurrentDirUTF8 : String
10513: Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10514: Function CreateDirUTF8( const NewDir : String) : Boolean
10515: Function RemoveDirUTF8( const Dir : String) : Boolean
10516: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10517: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10518: Function FileCreateUTF8( const FileName : string) : THandle;
10519: Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10520: Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10521: Function FileSizeUtf8( const Filename : string) : int64
10522: Function GetFileDescription( const AFilename : string) : string
10523: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10524: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10525: Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10526: Function IsUNCPath( const Path : String) : Boolean
10527: Function ExtractUNCVolume( const Path : String) : String
10528: Function ExtractFileRoot( FileName : String) : String
10529: Function GetDarwinSystemFilename( Filename : string) : string
10530: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10531: Function StrToCmdLineParam( const Param : string) : string
10532: Function MergeCmdLineParams( ParamList : TStrings) : string
10533: Procedure InvalidateFileStateCache( const Filename : string)
10534: Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10535: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10536: Function ReadFileToString( const Filename : string) : string
10537: type
10538:   TCopyFileFlag = ( cffOverwriteFile,
10539:     cffCreateDestDirectory, cffPreserveTime );
10540:   TCopyFileFlags = set of TCopyFileFlag;*)
10541:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10542:   TCopyFileFlags', 'set of TCopyFileFlag
10543: Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10544: end;
10545:
10546: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10547: begin
10548:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10549:   SIRegister_TMask(CL);
10550:   SIRegister_TParseStringList(CL);
10551:   SIRegister_TMaskList(CL);
10552: Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10553: Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10554: Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10555: Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10556: end;
10557:
10558: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10559: begin
10560:   //PShellHookInfo', '^TShellHookInfo // will not work
10561:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10562:   SHELLHOOKINFO', 'TShellHookInfo
10563:   LPSHELLHOOKINFO', 'PShellHookInfo
10564:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10565:   SIRegister_TJvShellHook(CL);
10566: Function InitJvShellHooks : Boolean
10567: Procedure UnInitJvShellHooks
10568: end;
10569:
10570: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10571: begin
10572:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10573:   +', dcHasSetSel, dcWantTab, dcNative )
10574:   TDlgCodes', 'set of TDlgCode
10575:   'dcWantMessage', '').SetString( dcWantAllKeys);
10576:   SIRegister_IJvExControl(CL);
10577:   SIRegister_IJvDenySubClassing(CL);
10578:   SIRegister_TStructPtrMessage(CL);
10579: Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10580: Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10581: Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10582: Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10583: Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10584: Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10585: Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10586: Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10587: Function GetFocusedControl( AControl : TControl) : TWinControl
10588: Function DlgCToDlgCodes( Value : Longint) : TDlgCodes

```

```

10589: Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10590: Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10591: Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10592:   SIRegister_TJvExControl(CL);
10593:   SIRegister_TJvExWinControl(CL);
10594:   SIRegister_TJvExCustomControl(CL);
10595:   SIRegister_TJvExGraphicControl(CL);
10596:   SIRegister_TJvExHintWindow(CL);
10597:   SIRegister_TJvExPubGraphicControl(CL);
10598: end;
10599:
10600: (*-----*)
10601: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10602: begin
10603:   Procedure EncodeStream( Input, Output : TStream)
10604:   Procedure DecodeStream( Input, Output : TStream)
10605:   Function EncodeString1( const Input : string) : string
10606:   Function DecodeString1( const Input : string) : string
10607: end;
10608:
10609: (*-----*)
10610: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10611: begin
10612:   SIRegister_TWebAppRegInfo(CL);
10613:   SIRegister_TWebAppRegList(CL);
10614:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10615:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10616:   Procedure UnregisterWebApp( const AProgID : string)
10617:   Function FindRegisteredWebApp( const AProgID : string) : string
10618:   Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10619:   'sUDPPort', 'String').SetString( 'UDPPort
10620: end;
10621:
10622: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10623: begin
10624:   // TStringDynArray, 'array of string
10625:   Function GetEnvVarValue( const VarName : string) : string
10626:   Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10627:   Function DeleteEnvVar( const VarName : string) : Integer
10628:   Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
10629:   BufSize:Int):Int;
10630:   Function ExpandEnvVars( const Str : string) : string
10631:   Function GetAllEnvVars( const Vars : TStrings) : Integer
10632:   Procedure GetAllEnvVarNames( const Names : TStrings);
10633:   Function GetAllEnvVarNames1 : TStringDynArray;
10634:   Function EnvBlockSize : Integer
10635:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10636:   SIRegister_TPJEnvVarsEnumerator(CL);
10637:   SIRegister_TPJEnvVars(CL);
10638:   FindClass('TObject'), 'EPJEnvVars
10639:   FindClass('TObject'), 'EPJEnvVars
10640:   //Procedure Register
10641: end;
10642: (*-----*)
10643: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10644: begin
10645:   'cOneSecInMS', 'LongInt').SetInt( 1000);
10646:   // 'cDefTimeSlice', 'LongInt').SetInt( 50);
10647:   // 'cDefMaxExecTime', '').SetString( cOneMinInMS);
10648:   'cAppErrorMask', 'LongInt').SetInt( 1 shl 29);
10649:   Function IsApplicationError( const ErrCode : LongWord) : Boolean
10650:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10651:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10652:   Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10653:   Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10654:   Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10655:   Function MakeSize( const ACX, ACY : LongInt) : TSize
10656:   SIRegister_TPJCustomConsoleApp(CL);
10657:   SIRegister_TPJConsoleApp(CL);
10658: end;
10659:
10660: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10661: begin
10662:   INVALID_IP_ADDRESS', 'LongWord').SetUInt( $ffffff);
10663:   t_encoding', '( uuencode, base64, mime )
10664:   Function internet_date( date : TDateTime) : string
10665:   Function lookup_hostname( const hostname : string) : longint
10666:   Function my_hostname : string
10667:   Function my_ip_address : longint
10668:   Function ip2string( ip_address : longint) : string
10669:   Function resolve_hostname( ip : longint) : string
10670:   Function address_from( const s : string; count : integer) : string
10671:   Function encode_base64( data : TStream) : TStringList
10672:   Function decode_base64( source : TStringList) : TMemoryStream
10673:   Function posn( const s, t : string; count : integer) : integer
10674:   Function poscn( c : char; const s : string; n : integer) : integer
10675:   Function filename_of( const s : string) : string
10676:   //Function trim( const s : string) : string

```

```

10677: //Procedure setlength( var s : string; l : byte)
10678: Function TimeZoneBias : longint
10679: Function eight2seven_quoteprint( const s : string) : string
10680: Function eight2seven_german( const s : string) : string
10681: Function seven2eight_quoteprint( const s : string) : string end;
10682: type in_addr, 'record s_bytes : array[1..4] of byte; end;
10683: Function socketerror : cint
10684: Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10685: Function fprecv( s : cint; buf : __pointer; len : size_t; flags : cint) : ssize_t
10686: Function fpsend( s : cint; msg : __pointer; len : size_t; flags : cint) : ssize_t
10687: //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10688: Function fplistn( s : cint; backlog : cint) : cint
10689: //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10690: //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10691: //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10692: Function NetAddrToStr( Entry : in_addr) : String
10693: Function HostAddrToStr( Entry : in_addr) : String
10694: Function StrToHostAddr( IP : String) : in_addr
10695: Function StrToNetAddr( IP : String) : in_addr
10696: SOL_SOCKET, 'LongWord').SetUInt( $ffff);
10697: cint8, 'shortint
10698: cuint8, 'byte
10699: cchar, 'cint8
10700: cschar, 'cint8
10701: cuchar, 'cuint8
10702: cint16, 'smallint
10703: cuint16, 'word
10704: cshort, 'cint16
10705: csshort, 'cint16
10706: cushort, 'cuint16
10707: cint32, 'longint
10708: cuint32, 'longword
10709: cint, 'cint32
10710: csint, 'cint32
10711: cuint, 'cuint32
10712: csigned, 'cint
10713: cunsigned, 'cuint
10714: cint64, 'int64
10715: clonglong, 'cint64
10716: cslonglong, 'cint64
10717: cbool, 'longbool
10718: cfloat, 'single
10719: cdouble, 'double
10720: clongdouble, 'extended
10721:
10722: procedure SIRegister_uLkJJSON(CL: TPSPascalCompiler);
10723: begin
10724:   TlkJSONTypes', '(jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
10725:   SIRegister_TlkJSONdotnetclass(CL);
10726:   SIRegister_TlkJSONbase(CL);
10727:   SIRegister_TlkJSONnumber(CL);
10728:   SIRegister_TlkJSONstring(CL);
10729:   SIRegister_TlkJSONboolean(CL);
10730:   SIRegister_TlkJSONnull(CL);
10731:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10732:   + 'se; data : TObjct; var Continue : Boolean)
10733:   SIRegister_TlkJSONcustomlist(CL);
10734:   SIRegister_TlkJSONlist(CL);
10735:   SIRegister_TlkJSONobjectmethod(CL);
10736:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10737:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10738:   SIRegister_TlkHashTable(CL);
10739:   SIRegister_TlkBalTree(CL);
10740:   SIRegister_TlkJSONobject(CL);
10741:   SIRegister_TlkJSON(CL);
10742:   SIRegister_TlkJSONstreamed(CL);
10743:   Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer) : string
10744: end;
10745:
10746: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10747: begin
10748:   TZListSortCompare', 'Function (Item1, Item2 : TObjct) : Integer
10749:   SIRegister_TZSortedList(CL);
10750:   Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10751:   Function zLastDelimiter( const Delimiters, Str : string) : Integer
10752:   //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10753:   //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10754:   Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10755:   Function zStartsWith( const Str, SubStr : RawByteString) : Boolean;
10756:   Function zEndsWith( const Str, SubStr : WideString) : Boolean;
10757:   Function zEndsWith( const Str, SubStr : RawByteString) : Boolean;
10758:   Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10759:   Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10760:   Function SQLStrToFloat( const Str : AnsiString) : Extended
10761:   //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10762:   //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10763:   Function BufferToBytes( Buffer : TObjct; Length : LongInt) : TByteDynArray
10764:   Function StrToBoolEx( Str : string) : Boolean
10765:   Function BoolToStrEx( Bool : Boolean) : String

```



```

10766: Function IsIpAddr( const Str : string ) : Boolean
10767: Function zSplitString( const Str, Delimiters : string ) : TStrings
10768: Procedure PutSplitString( List : TStrings; const Str, Delimiters : string )
10769: Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string )
10770: Function ComposeString( List : TStrings; const Delimiter : string ) : string
10771: Function FloatToSQLStr( Value : Extended ) : string
10772: Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string )
10773: Function SplitStringEx( const Str, Delimiter : string ) : TStrings
10774: Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string )
10775: Function zBytesToStr( const Value : TByteDynArray ) : AnsiString
10776: Function zStrToBytes( const Value : AnsiString ) : TByteDynArray;
10777: Function StrToBytes1( const Value : UTF8String ) : TByteDynArray;
10778: Function StrToBytes2( const Value : RawByteString ) : TByteDynArray;
10779: Function StrToBytes3( const Value : WideString ) : TByteDynArray;
10780: Function StrToBytes4( const Value : UnicodeString ) : TByteDynArray;
10781: Function BytesToVar( const Value : TByteDynArray ) : Variant
10782: Function VarToBytes( const Value : Variant ) : TByteDynArray
10783: Function AnsiSQLDateToDateTime( const Value : string ) : TDateTime
10784: Function TimestampStrToDateTime( const Value : string ) : TDateTime
10785: Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean ) : string
10786: Function EncodeCString( const Value : string ) : string
10787: Function DecodeCString( const Value : string ) : string
10788: Function zReplaceChar( const Source, Target : Char; const Str : string ) : string
10789: Function MemPas( Buffer : PChar; Length : LongInt ) : string
10790: Procedure DecodesQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
SubVersion:Int);
10791: Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Integer):Int;
10792: Function FormatSQLVersion( const SQLVersion : Integer ) : String
10793: Function ZStrToFloat( Value : AnsiChar ) : Extended;
10794: Function ZStrToFloat1( Value : AnsiString ) : Extended;
10795: Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10796: Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10797: Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10798: Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10799: Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10800: Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10801: Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10802: end;
10803:
10804: unit uPSI_ZEncoding;
10805: Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word ) : RawByteString
10806: Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word ) : String
10807: Function ZRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
10808: Function ZUnicodeToRaw( const US : WideString; CP : Word ) : RawByteString
10809: Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word ) : RawByteString
10810: Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word ) : AnsiString
10811: Function ZConvertAnsiToUTF8( const Src : AnsiString ) : UTF8String
10812: Function ZConvertUTF8ToAnsi( const Src : UTF8String ) : AnsiString
10813: Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
10814: Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString
10815: Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word ) : String
10816: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word ) : RawByteString
10817: Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10818: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word ) : String
10819: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word ) : UTF8String
10820: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP : Word ) : UTF8String
10821: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word ) : AnsiString
10822: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP : Word ) : AnsiString
10823: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word ) : String
10824: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word ) : String
10825: Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word ) : String
10826: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word ) : WideString
10827: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word ) : WideString
10828: Function ZConvertStringToUnicodeWithAutoEncode( const Src : String; const StringCP:Word):WideString
10829: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word ) : RawByteString
10830: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word ) : AnsiString
10831: Function ZMoveAnsiToUTF8( const Src : AnsiString ) : UTF8String
10832: Function ZMoveUTF8ToAnsi( const Src : UTF8String ) : AnsiString
10833: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
10834: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString
10835: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word ) : AnsiString
10836: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word ) : String
10837: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word ) : String
10838: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word ) : RawByteString
10839: Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word ) : String
10840: Function ZMoveStringToUTF8( const Src : String; const StringCP : Word ) : UTF8String
10841: Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
10842: Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word ) : WideString
10843: Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word ) : RawByteString
10844: Function ZDefaultSystemCodePage : Word
10845: Function ZCompatibleCodePages( const CP1, CP2 : Word ) : Boolean
10846:
10847:
10848: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10849: begin
10850:   'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt').SetInt( 0);
10851:   ('RPC_C_AUTHN_LEVEL_NONE','LongInt').SetInt( 1);
10852:   ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt').SetInt( 2);

```

```

10853: ('RPC_C_AUTHN_LEVEL_CALL', 'LongInt').SetInt( 3);
10854: ('RPC_C_AUTHN_LEVEL_PKT', 'LongInt').SetInt( 4);
10855: ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY', 'LongInt').SetInt( 5);
10856: ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY', 'LongInt').SetInt( 6);
10857: (('alDefault', '1').SetString( RPC_C_AUTHN_LEVEL_DEFAULT);
10858: ('alNone', '2').SetString( RPC_C_AUTHN_LEVEL_NONE);
10859: ('alConnect', '3').SetString( RPC_C_AUTHN_LEVEL_CONNECT);
10860: ('alCall', '4').SetString( RPC_C_AUTHN_LEVEL_CALL);
10861: ('alPacket', '5').SetString( RPC_C_AUTHN_LEVEL_PKT);
10862: ('alPacketIntegrity', '6').SetString( RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10863: ('alPacketPrivacy', '7').SetString( RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10864: ('RPC_C_IMP_LEVEL_DEFAULT', 'LongInt').SetInt( 0);
10865: ('RPC_C_IMP_LEVEL_ANONYMOUS', 'LongInt').SetInt( 1);
10866: ('RPC_C_IMP_LEVEL_IDENTIFY', 'LongInt').SetInt( 2);
10867: ('RPC_C_IMP_LEVEL_IMPERSONATE', 'LongInt').SetInt( 3);
10868: ('RPC_C_IMP_LEVEL_DELEGATE', 'LongInt').SetInt( 4);
10869: (('ilDefault', '0').SetString( RPC_C_IMP_LEVEL_DEFAULT);
10870: ('ilAnonymous', '1').SetString( RPC_C_IMP_LEVEL_ANONYMOUS);
10871: ('ilIdentiry', '2').SetString( RPC_C_IMP_LEVEL_IDENTIFY);
10872: ('ilImpersonate', '3').SetString( RPC_C_IMP_LEVEL_IMPERSONATE);
10873: ('ilDelegate', '4').SetString( RPC_C_IMP_LEVEL_DELEGATE);}
10874: ('EOAC_NONE', 'LongWord').SetUInt( $0);
10875: ('EOAC_DEFAULT', 'LongWord').SetUInt( $800);
10876: ('EOAC_MUTUAL_AUTH', 'LongWord').SetUInt( $1);
10877: ('EOAC_STATIC_CLOACKING', 'LongWord').SetUInt( $20);
10878: ('EOAC_DYNAMIC_CLOACKING', 'LongWord').SetUInt( $40);
10879: ('EOAC_ANY_AUTHORITY', 'LongWord').SetUInt( $80);
10880: ('RPC_C_AUTHN_WINNT', 'LongInt').SetInt( 10);
10881: ('RPC_C_AUTHNZ_NONE', 'LongInt').SetInt( 0);
10882: ('RPC_C_AUTHNZ_NAME', 'LongInt').SetInt( 1);
10883: ('RPC_C_AUTHNZ_DCE', 'LongInt').SetInt( 2);
10884: FindClass('TOBJECT'), 'EBoIdCom
10885: Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10886: Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10887: Function BoldStreamToVariant( Stream : TStream) : OleVariant
10888: Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10889: Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10890: Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10891: Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10892: Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10893: Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10894: Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10895: Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10896: Function BoldCreateGUID : TGUID
10897: Function BoldCreateComObject( const ClsId, IID : TGUID; out Obj : variant; out Res : HRESULT) : Boolean
10898: Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IID:TGUID;out Obj:variant;out
Res:HRESULT):Bool;
10899: Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10900: Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10901: end;
10902:
10903: (*-----*)
10904: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10905: begin
10906:   Function ParseISODate( s : string) : TDateTime
10907:   Function ParseISODateTime( s : string) : TDateTime
10908:   Function ParseISOTime( str : string) : TDateTime
10909: end;
10910:
10911: (*-----*)
10912: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10913: begin
10914:   Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10915:   Function BoldCreateGUIDWithBracketsAsString : string
10916: end;
10917:
10918: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10919: begin
10920:   FindClass('TOBJECT'), 'TBoIdFileHandler
10921:   FindClass('TOBJECT'), 'TBoIdDiskFileHandler
10922:   //TBoIdFileHandlerClass', 'class of TBoIdFileHandler
10923:   TBoIdInitializeFileContents', 'Procedure ( StringList : TStringList)
10924:   SIRegister_TBoldFileHandler(CL);
10925:   SIRegister_TBoldDiskFileHandler(CL);
10926:   Procedure BoldCloseAllFilehandlers
10927:   Procedure BoldRemoveUnchangedFilesFromEditor
10928:   Function BoldFileHandlerList : TBoIdObjectArray
10929:   Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoIdModuleType;ShowInEditor:Bool;
OnInitializeFileContents : TBoIdInitializeFileContents) : TBoIdFileHandler
10930: end;
10931:
10932: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10933: begin
10934:   PCharArr', 'array of PChar
10935:   Function BoldInternetOpen(Agent:String;
AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
10936:   Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags;Context:cardinal):Pointer
10937:   Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumberOfBytesToRead:Card;var
NumberOfBytesRead:Card):LongBool;

```

```

10938: Function BoldInternetCloseHandle( HINet : Pointer ) : LongBool
10939: Function BoldHttpRequestInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
Cardinal; Reserved : Cardinal ) : LongBool
10940: Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
Cardinal; Context : Cardinal ) : LongBool
10941: Function BoldHttpRequestOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
: PCharArr; Flags, Context : Cardinal ) : Pointer
10942: Function BoldHttpRequestSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
10943: Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
10944: Function BoldInternetAttemptConnect( dwReserved : DWORD ) : DWORD
10945: Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
10946: Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
10947: end;
10948:
10949: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
10950: begin
10951:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
10952:   SIRegister_TfrmBoldQueryUser(CL);
10953:   Function QueryUser( const Title, Query : string ) : TBoldQueryResult
10954: end;
10955:
10956: (*-----*)
10957: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
10958: begin
10959:   (('befIsInDisplayList','').SetString( BoldElementFlag0);
10960:   (('befStronglyDependedOfPrioritized','').SetString( BoldElementFlag1);
10961:   (('befFollowerSelected','').SetString( BoldElementFlag2);
10962:   FindClass('TOBJECT'),'TBoldQueue
10963:   FindClass('TOBJECT'),'TBoldQueueable
10964:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
10965:   SIRegister_TBoldQueueable(CL);
10966:   SIRegister_TBoldQueue(CL);
10967:   Function BoldQueueFinalized : Boolean
10968:   Function BoldInstalledQueue : TBoldQueue
10969: end;
10970:
10971: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
10972: begin
10973:   const mmPerInch,'Extended').setExtended( 25.4);
10974:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,
10975:   + ' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc
10976:   + 'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda
10977:   + 'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC
10978:   + 'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
10979:   TBarLineType', '( white, black, black_half )
10980:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
10981:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st
10982:   + 'pBottomLeft, stpBottomRight, stpBottomCenter )
10983:   TCheckSumMethod', '( csmNone, csmModulo10 )
10984:   SIRegister_TAsBarcode(CL);
10985:   Function CheckSumModulo10( const data : string ) : string
10986:   Function ConvertMmToPixelsX( const Value : Double ) : Integer
10987:   Function ConvertMmToPixelsY( const Value : Double ) : Integer
10988:   Function ConvertInchToPixelsX( const Value : Double ) : Integer
10989:   Function ConvertInchToPixelsY( const Value : Double ) : Integer
10990: end;
10991:
10992: procedure SIRegister_Geometry(CL: TPSPascalCompiler); //OpenGL
10993: begin
10994:   THomogeneousByteVector', 'array[0..3] of Byte
10995:   THomogeneousWordVector', 'array[0..3] of Word
10996:   THomogeneousIntVector', 'array[0..3] of Integer
10997:   THomogeneousFltVector', 'array[0..3] of single
10998:   THomogeneousDblVector', 'array[0..3] of double
10999:   THomogeneousExtVector', 'array[0..3] of extended
11000:   TAffineByteVector', 'array[0..2] of Byte
11001:   TAffineWordVector', 'array[0..2] of Word
11002:   TAffineIntVector', 'array[0..2] of Integer
11003:   TAffineFltVector', 'array[0..2] of single
11004:   TAffineDblVector', 'array[0..2] of double
11005:   TAffineExtVector', 'array[0..2] of extended
11006:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11007:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11008:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11009:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11010:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11011:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11012:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11013:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11014:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11015:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11016:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11017:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11018:   TMatrix4b', 'THomogeneousByteMatrix
11019:   TMatrix4w', 'THomogeneousWordMatrix
11020:   TMatrix4i', 'THomogeneousIntMatrix
11021:   TMatrix4f', 'THomogeneousFltMatrix
11022:   TMatrix4d', 'THomogeneousDblMatrix

```

```

11023: TMatrix4e', 'THomogeneousExtMatrix
11024: TMatrix3b', 'TAffineByteMatrix
11025: TMatrix3w', 'TAffineWordMatrix
11026: TMatrix3i', 'TAffineIntMatrix
11027: TMatrix3f', 'TAffineFltMatrix
11028: TMatrix3d', 'TAffineDblMatrix
11029: TMatrix3e', 'TAffineExtMatrix
11030: ///PMatrix, '^TMatrix // will not work
11031: TMatrixGL', 'THomogeneousFltMatrix
11032: THomogeneousMatrix', 'THomogeneousFltMatrix
11033: TAffineMatrix', 'TAffineFltMatrix
11034: TQuaternion', record Vector : TVector4f; end
11035: TRectangle', record Left : integer; Top : integer; Width : inte'
11036: ger; Height : Integer; end
11037: TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11038:   + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11039:   + ', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11040: 'EPSILON', 'Extended').setExtended( 1E-100);
11041: 'EPSILON2', 'Extended').setExtended( 1E-50);
11042: Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11043: Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11044: Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11045: Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11046: Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11047: Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11048: Function VectorAngle( V1, V2 : TAffineVector) : Single
11049: Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11050: Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11051: Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11052: Function VectorLength( V : array of Single) : Single
11053: Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11054: Procedure VectorNegate( V : array of Single)
11055: Function VectorNorm( V : array of Single) : Single
11056: Function VectorNormalize( V : array of Single) : Single
11057: Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11058: Function VectorReflect( V, N : TAffineVector) : TAffineVector
11059: Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11060: Procedure VectorScale( V : array of Single; Factor : Single)
11061: Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11062: Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11063: Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11064: Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11065: Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11066: Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11067: Procedure MatrixAdjoint( var M : TMatrixGL)
11068: Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11069: Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11070: Function MatrixDeterminant( M : TMatrixGL) : Single
11071: Procedure MatrixInvert( var M : TMatrixGL)
11072: Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11073: Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11074: Procedure MatrixTranspose( var M : TMatrixGL)
11075: Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11076: Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11077: Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11078: Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11079: Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11080: Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11081: Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11082: Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11083: ///Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11084: Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11085: Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11086: Function VectorTransforml( V : TVector3f; M : TMatrixGL) : TVector3f;
11087: Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11088: Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11089: Function MakeAffineVector( V : array of Single) : TAffineVector
11090: Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11091: Function MakeVector( V : array of Single) : TVectorGL
11092: Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11093: Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11094: Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11095: Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11096: Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11097: Function ArcCosGL( X : Extended) : Extended
11098: Function ArcSinGL( X : Extended) : Extended
11099: Function ArcTan2GL( Y, X : Extended) : Extended
11100: Function CoTanGL( X : Extended) : Extended
11101: Function DegToRadGL( Degrees : Extended) : Extended
11102: Function RadToDegGL( Radians : Extended) : Extended
11103: Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11104: Function TanGL( X : Extended) : Extended
11105: Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11106: Function Turnl( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11107: Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11108: Function Pitchl( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11109: Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11110: Function Rolll( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11111: end;

```



```

11112:
11113:
11114: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11115: begin
11116:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string ) : Longint
11117:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string ) : Boolean
11118:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string ) : Boolean
11119:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string ) : Boolean
11120:   Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean ) : Boolean
11121:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string ) : Integer
11122:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer ) : Integer
11123:   Function RegReadString( const RootKey : HKEY; const Key, Name : string ) : string
11124:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string ) : string
11125:   Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string ) : Int64
11126:   Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64 ) : Int64
11127:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean )
11128:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer )
11129:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string )
11130:   Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64 )
11131:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStringList ) : Boolean
11132:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStringList ) : Boolean
11133:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string ) : Boolean
11134:   Function RegKeyExists( const RootKey : HKEY; const Key : string ) : Boolean
11135:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11136:   + 'RunOnce, ekServiceRun, ekServiceRunOnce )
11137:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11138:   Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string ) : Boolean
11139:   Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string ) : Boolean
11140:   Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
Items:TStrings):Bool;
11141:   Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
SaveTo:TStrings):Bool;
11142:   Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string) : Boolean
11143: end;
11144:
11145: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11146: begin
11147:   CLSID_StdComponentCategoriesMgr, 'TGUID').SetString( '{0002E005-0000-0000-C000-000000000046}'
11148:   CATID_SafeForInitializing, 'TGUID').SetString( '{7DD95802-9882-11CF-9FA9-00AA006C42C4}'
11149:   CATID_SafeForScripting, 'TGUID').SetString( '{7DD95801-9882-11CF-9FA9-00AA006C42C4}'
11150:   icMAX_CATEGORY_DESC_LEN, 'LongInt').SetInt( 128);
11151:   FindClass('TOBJECT'),'EInvalidParam
11152:   Function IsDCOMInstalled : Boolean
11153:   Function IsDCOMEnabled : Boolean
11154:   Function GetDCOMVersion : string
11155:   Function GetMDACVersion : string
11156:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11157:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11158:   Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11159:   Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11160:   Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11161:   Function CreateComponentCategory( const CatID : TGUID; const sDescription : string ) : HResult
11162:   Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HResult
11163:   Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HResult
11164:   Function ResetIStreamToStart( Stream : IStream ) : Boolean
11165:   Function SizeOfIStreamContents( Stream : IStream ) : Largeint
11166:   Function StreamToVariantArray( Stream : TStream ) : OleVariant;
11167:   Function StreamToVariantArray1( Stream : IStream ) : OleVariant;
11168:   Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11169:   Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11170: end;
11171:
11172:
11173: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11174: begin
11175:   Const('CelsiusFreezingPoint', 'Extended').setExtended( 0.0);
11176:   FahrenheitFreezingPoint, 'Extended').setExtended( 32.0);
11177:   KelvinFreezingPoint, 'Extended').setExtended( 273.15);
11178:   CelsiusAbsoluteZero, 'Extended').setExtended( - 273.15);
11179:   FahrenheitAbsoluteZero, 'Extended').setExtended( - 459.67);
11180:   KelvinAbsoluteZero, 'Extended').setExtended( 0.0);
11181:   DegPerCycle, 'Extended').setExtended( 360.0);
11182:   DegPerGrad, 'Extended').setExtended( 0.9);
11183:   DegPerRad, 'Extended').setExtended( 57.295779513082320876798154814105);
11184:   GradPerCycle, 'Extended').setExtended( 400.0);
11185:   GradPerDeg, 'Extended').setExtended( 1.11111111111111111111111111111111);
11186:   GradPerRad, 'Extended').setExtended( 63.661977236758134307553505349006);
11187:   RadPerCycle, 'Extended').setExtended( 6.283185307179586476925286766559);
11188:   RadPerDeg, 'Extended').setExtended( 0.017453292519943295769236907684886);
11189:   RadPerGrad, 'Extended').setExtended( 0.015707963267948966192313216916398);
11190:   CyclePerDeg, 'Extended').setExtended( 0.0027777777777777777777777777777778);
11191:   CyclePerGrad, 'Extended').setExtended( 0.0025);
11192:   CyclePerRad, 'Extended').setExtended( 0.15915494309189533576888376337251);
11193:   ArcMinutesPerDeg, 'Extended').setExtended( 60.0);
11194:   ArcSecondsPerArcMinute, 'Extended').setExtended( 60.0);
11195:   Function HowAOneLinerCanBiteYou( const Step, Max : Longint ) : Longint

```

```

11196: Function MakePercentage( const Step, Max : Longint ) : Longint
11197: Function CelsiusToKelvin( const T : double ) : double
11198: Function CelsiusToFahrenheit( const T : double ) : double
11199: Function KelvinToCelsius( const T : double ) : double
11200: Function KelvinToFahrenheit( const T : double ) : double
11201: Function FahrenheitToCelsius( const T : double ) : double
11202: Function FahrenheitToKelvin( const T : double ) : double
11203: Function CycleToDeg( const Cycles : double ) : double
11204: Function CycleToGrad( const Cycles : double ) : double
11205: Function CycleToRad( const Cycles : double ) : double
11206: Function DegToCycle( const Degrees : double ) : double
11207: Function DegToGrad( const Degrees : double ) : double
11208: Function DegToRad( const Degrees : double ) : double
11209: Function GradToCycle( const Grads : double ) : double
11210: Function GradToDeg( const Grads : double ) : double
11211: Function GradToRad( const Grads : double ) : double
11212: Function RadToCycle( const Radians : double ) : double
11213: Function RadToDeg( const Radians : double ) : double
11214: Function RadToGrad( const Radians : double ) : double
11215: Function DmsToDeg( const D, M : Integer; const S : double ) : double
11216: Function DmsToRad( const D, M : Integer; const S : double ) : double
11217: Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11218: Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal ) : string
11219: Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11220: Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11221: Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11222: Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11223: Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11224: Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11225: Function CmToInch( const Cm : double ) : double
11226: Function InchToCm( const Inch : double ) : double
11227: Function FeetToMetre( const Feet : double ) : double
11228: Function MetreToFeet( const Metre : double ) : double
11229: Function YardToMetre( const Yard : double ) : double
11230: Function MetreToYard( const Metre : double ) : double
11231: Function NmToKm( const Nm : double ) : double
11232: Function KmToNm( const Km : double ) : double
11233: Function KmToSm( const Km : double ) : double
11234: Function SmToKm( const Sm : double ) : double
11235: Function LitreToGalUs( const Litre : double ) : double
11236: Function GalUsToLitre( const GalUs : double ) : double
11237: Function GalUsToGalCan( const GalUs : double ) : double
11238: Function GalCanToGalUs( const GalCan : double ) : double
11239: Function GalUsToGalUk( const GalUs : double ) : double
11240: Function GalUkToGalUs( const GalUk : double ) : double
11241: Function LitreToGalCan( const Litre : double ) : double
11242: Function GalCanToLitre( const GalCan : double ) : double
11243: Function LitreToGalUk( const Litre : double ) : double
11244: Function GalUkToLitre( const GalUk : double ) : double
11245: Function KgToLb( const Kg : double ) : double
11246: Function LbToKg( const Lb : double ) : double
11247: Function KgToOz( const Kg : double ) : double
11248: Function OzToKg( const Oz : double ) : double
11249: Function CwtUsToKg( const Cwt : double ) : double
11250: Function CwtUkToKg( const Cwt : double ) : double
11251: Function KaratToKg( const Karat : double ) : double
11252: Function KgToCwtUs( const Kg : double ) : double
11253: Function KgToCwtUk( const Kg : double ) : double
11254: Function KgToKarat( const Kg : double ) : double
11255: Function KgToSton( const Kg : double ) : double
11256: Function KgToLton( const Kg : double ) : double
11257: Function StonToKg( const STon : double ) : double
11258: Function LtonToKg( const Lton : double ) : double
11259: Function QrUsToKg( const Qr : double ) : double
11260: Function QrUkToKg( const Qr : double ) : double
11261: Function KgToQrUs( const Kg : double ) : double
11262: Function KgToQrUk( const Kg : double ) : double
11263: Function PascalToBar( const Pa : double ) : double
11264: Function PascalToAt( const Pa : double ) : double
11265: Function PascalToTorr( const Pa : double ) : double
11266: Function BarToPascal( const Bar : double ) : double
11267: Function AtToPascal( const At : double ) : double
11268: Function TorrToPascal( const Torr : double ) : double
11269: Function KnotToMs( const Knot : double ) : double
11270: Function HpElectricToWatt( const HpE : double ) : double
11271: Function HpMetricToWatt( const HpM : double ) : double
11272: Function MsToKnot( const ms : double ) : double
11273: Function WattToHpElectric( const W : double ) : double
11274: Function WattToHpMetric( const W : double ) : double
11275: function getBigPI: string; //PI of 1000 numbers
11276:
11277: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11278: begin
11279:   Function CDEExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
11280:   Procedure CDCopyFile( const FileName, DestName : string)
11281:   Procedure CDMoveFile( const FileName, DestName : string)
11282:   Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor ) : TColor
11283:   Procedure CDDeleteFiles( Sender : TObject; s : string)
11284:   Function CDGetTempDir : string

```

```

11285: Function CDGetFileSize( FileName : string) : longint
11286: Function GetFileTime( FileName : string) : longint
11287: Function GetShortName( FileName : string) : string
11288: Function GetFullName( FileName : string) : string
11289: Function WinReboot : boolean
11290: Function WinDir : String
11291: Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11292: Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11293: Function devExecutor : TdevExecutor
11294: end;
11295:
11296: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11297: begin
11298:   Procedure CheckAssociations // AssociationsCount', 'LongInt').SetInt( 7);
11299:   Procedure Associate( Index : integer)
11300:   Procedure UnAssociate( Index : integer)
11301:   Function IsAssociated( Index : integer) : boolean
11302:   Function CheckFileType( const extension, filetype, description, verb, serverapp : string) : boolean
11303:   Procedure RegisterFileType( const extension, filetype, description, verb, serverapp, IcoNum: string)
11304:   Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11305:   procedure RefreshIcons;
11306:   function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11307:   function MergColor(Colors: Array of TColor): TColor;
11308:   function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11309:   procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11310:   function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11311:   function GetInverseColor(AColor: TColor): TColor;
11312:   procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11313:   procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X, Y: integer; ShadowColor: TColor);
11314:   procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11315:   Procedure GetSystemMenuFont(Font: TFont);
11316: end;
11317:
11318: //*****unit uPSI_JvHLParse;*****
11319: function IsStringConstant(const St: string): Boolean;
11320: function IsIntConstant(const St: string): Boolean;
11321: function IsRealConstant(const St: string): Boolean;
11322: function IsIdentifier(const ID: string): Boolean;
11323: function GetStringValue(const St: string): string;
11324: procedure ParseString(const S: string; Ss: TStrings);
11325: function IsStringConstantW(const St: WideString): Boolean;
11326: function IsIntConstantW(const St: WideString): Boolean;
11327: function IsRealConstantW(const St: WideString): Boolean;
11328: function IsIdentifierW(const ID: WideString): Boolean;
11329: function GetStringValueW(const St: WideString): WideString;
11330: procedure ParseStringW(const S: WideString; Ss: TStrings);
11331:
11332:
11333: //*****unit uPSI_JclMapi;*****
11334:
11335: Function JclSimpleSendMail( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
  TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11336: Function JclSimpleSendFax( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
  TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11337: Function JclSimpleBringUpSendMailDialog(const ASubject, ABody: string; const
  AAttach: TFileName; AParentWND: HWND): Bool
11338: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11339: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11340:
11341: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11342: begin
11343:   // 'Pdes_key_schedule', '^des_key_schedule // will not work
11344:   Function BuildType1Message( ADomain, AHost : String) : String
11345:   Function BuildType3Message(ADomain, AHost, AUsername: WideString; APassword, ANonce: String): String
11346:   Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11347:   Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11348:   GBase64CodeTable, 'string').SetString(' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11349:   GXXECodeTable', 'string').SetString('+-0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11350:   GUUECodeTable', 'string').SetString('!"#$%&'()*+,-./0123456789;=<?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11351: end;
11352:
11353: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11354: begin
11355:   ('IpAny', 'LongWord').SetUInt( $00000000);
11356:   IpLoopBack, 'LongWord').SetUInt( $7F000001);
11357:   IpBroadcast, 'LongWord').SetUInt( $FFFFFFF);
11358:   IpNone, 'LongWord').SetUInt( $FFFFFFF);
11359:   PortAny, 'LongWord').SetUInt( $0000);
11360:   SocketMaxConnections, 'LongInt').SetInt( 5);
11361:   TIpAddr, 'LongWord
11362:   TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11363:   Function HostToNetLong( HostLong : LongWord) : LongWord
11364:   Function HostToNetShort( HostShort : Word) : Word
11365:   Function NetToHostLong( NetLong : LongWord) : LongWord
11366:   Function NetToHostShort( NetShort : Word) : Word
11367:   Function StrToIp( Ip : string) : TIpAddr
11368:   Function IpToStr( Ip : TIpAddr) : string
11369: end;
11370:

```

```

11371: (*-----*)
11372: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11373: begin
11374:   TAlsmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11375:   + 'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11376:   + 'amShal, AlsmtpClientAuthAutoSelect )
11377:   TAlsmtpClientAuthTypeSet', 'set of TAlsmtpClientAuthType
11378:   SIRegister_TAlsmtpClient(CL);
11379: end;
11380:
11381: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11382: begin
11383:   'TBitNo', 'Integer
11384:   TStByteNo', 'Integer
11385:   TStationNo', 'Integer
11386:   TInOutNo', 'Integer
11387:   TIO', '( EE, AA, NE, NA )
11388:   TBitSet', 'set of TBitNo
11389:   TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11390:   TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11391:   TBitAddr', 'LongInt
11392:   TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11393:   TByteAddr', 'SmallInt
11394:   TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11395:   Function BitAddr(aIo: TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11396:   Function BusBitAddr(aIo:TIO;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11397:   Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte;var
aBitNo:TBitNo);
11398:   Function BitAddrToStr( Value : TBitAddr) : string
11399:   Function StrToBitAddr( const Value : string) : TBitAddr
11400:   Function ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11401:   Function BusByteAddr(aIo:TIO;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11402:   Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte)
11403:   Function ByteAddrToStr( Value : TByteAddr) : string
11404:   Function StrToByteAddr( const Value : string) : TByteAddr
11405:   Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11406:   Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11407:   Function InOutStateToStr( State : TInOutState) : string
11408:   Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11409:   Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11410: end;
11411:
11412: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11413: begin
11414:   TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11415:   + 'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11416:   DpmiPmVector', 'Int64
11417:   'DInterval','LongInt').SetInt( 1000);
11418:   //'DEnabled','Boolean')BoolToStr( True);
11419:   'DIntFreq','string').SetString( if64
11420:   //'DMessages','Boolean').SetString( if64);
11421:   SIRegister_TwdxCustomTimer(CL);
11422:   SIRegister_TwdxTimer(CL);
11423:   SIRegister_TwdxRtcTimer(CL);
11424:   SIRegister_TCustomIntTimer(CL);
11425:   SIRegister_TIntTimer(CL);
11426:   SIRegister_TRtcIntTimer(CL);
11427:   Function RealNow : TDateTime
11428:   Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11429:   Function DateTimeToMs( Time : TDateTime) : LongInt
11430: end;
11431:
11432: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11433: begin
11434:   TIdSyslogPRI', 'Integer
11435:   TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11436:   + 'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11437:   + 'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11438:   + 'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11439:   + 'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11440:   TIdSyslogSeverity', '(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11441:   SIRegister_TIdSysLogMsgPart(CL);
11442:   SIRegister_TIdSysLogMessage(CL);
11443:   Function FacilityToString( AFac : TIdSyslogFacility) : string
11444:   Function SeverityToString( ASec : TIdSyslogSeverity) : string
11445:   Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11446:   Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11447:   Function NoToFacility( AFac : Word) : TIdSyslogFacility
11448:   Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11449: end;
11450:
11451: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11452: begin
11453:   'UWhitespace','String').SetString( '(?:\s*)
11454:   Function StripSpaces( const AText : string) : string
11455:   Function CharCount( const AText : string; Ch : Char) : Integer
11456:   Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11457:   Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11458: end;

```



```

11459:
11460:
11461: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11462: begin
11463:   ExtPascalVersion', 'String')).SetString( '0.9.8
11464:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11465:   + 'Opera, brKonqueror, brMobileSafari )
11466:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11467:   AddTypeS('TExtProcedure', 'Procedure
11468:   Function DetermineBrowser( const UserAgentStr : string ) : TBrowser
11469:   Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11470:   Function ExtExplode( Delim : char; const S : string; Separator : char ) : TStringList
11471:   Function FirstDelimiter( const Delimiters, S : string; Offset : integer ) : integer
11472:   Function RPosEx( const Substr, Str : string; Offset : integer ) : integer
11473:   Function CountStr( const Substr, Str : string; UntilStr : string ) : integer
11474:   Function StrToJS( const S : string; UseBR : boolean ) : string
11475:   Function CaseOf( const S : string; const Cases : array of string ) : integer
11476:   Function RCaseOf( const S : string; const Cases : array of string ) : integer
11477:   Function EnumToJSSString( TypeInfo : PTypeInfo; Value : integer ) : string
11478:   Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11479:   Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11480:   Function ExtBefore( const BeforeS, AfterS, S : string ) : boolean
11481:   Function IsUpperCase( S : string ) : boolean
11482:   Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11483:   Function BeautifyCSS( const AStyle : string ) : string
11484:   Function LengthRegExp( Rex : string; CountAll : Boolean ) : integer
11485:   Function JSDateToDateTime( JSDate : string ) : TDateTime
11486: end;
11487:
11488: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11489: begin
11490:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11491:   TSHDeleteOptions', 'set of TSHDeleteOption
11492:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11493:   TSHRenameOptions', 'set of TSHRenameOption
11494:   Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions ) : Boolean
11495:   Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions ) : Boolean
11496:   Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions ) : Boolean
11497:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11498:   TEnumFolderFlags', 'set of TEnumFolderFlag
11499:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWORD'
11500:   + 'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11501:   + 'TEnumIdList; Folder : IShellFolder; end
11502:   Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11503:   Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11504:   Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11505:   Function SHEnumFolderNext( var F : TEnumFolderRec ) : Boolean
11506:   Function GetSpecialFolderLocation( const Folder : Integer ) : string
11507:   Function DisplayPropDialog( const Handle : HWND; const FileName : string ) : Boolean;
11508:   Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList ) : Boolean;
11509:   Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint ) : Boolean
11510:   Function OpenFolder( const Path : string; Parent : HWND ) : Boolean
11511:   Function OpenSpecialFolder( FolderID : Integer; Parent : HWND ) : Boolean
11512:   Function SHReallocMem( var P : Pointer; Count : Integer ) : Boolean
11513:   Function SHAllocMem( out P : Pointer; Count : Integer ) : Boolean
11514:   Function SHGetMem( var P : Pointer; Count : Integer ) : Boolean
11515:   Function SHFreeMem( var P : Pointer ) : Boolean
11516:   Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder ) : PItemIdList
11517:   Function PathToPidl( const Path : string; Folder : IShellFolder ) : PItemIdList
11518:   Function PathToPidlBind( const FileName : string; out Folder : IShellFolder ) : PItemIdList
11519:   Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11520:   Function PidlCompare( const Pidl1, Pidl2 : PItemIdList ) : Boolean
11521:   Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList ) : Boolean
11522:   Function PidlFree( var IdList : PItemIdList ) : Boolean
11523:   Function PidlGetDepth( const Pidl : PItemIdList ) : Integer
11524:   Function PidlGetLength( const Pidl : PItemIdList ) : Integer
11525:   Function PidlGetNext( const Pidl : PItemIdList ) : PItemIdList
11526:   Function PidlToPath( IdList : PItemIdList ) : string
11527:   Function StrRetFreeMem( StrRet : TStrRet ) : Boolean
11528:   Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean ) : string
11529:   PShellLink', '^TShellLink // will not work
11530:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11531:   + 'ingDirectory : string; IdList : PItemIdList; Target : string; Description '
11532:   + ' : string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11533:   Procedure ShellLinkFree( var Link : TShellLink)
11534:   Function ShellLinkResolve( const FileName : string; var Link : TShellLink ) : HRESULT
11535:   Function ShellLinkCreate( const Link : TShellLink; const FileName : string ) : HRESULT
11536:   Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11537:   Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon ) : Boolean
11538:   Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo ) : Boolean
11539:   Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal ) : HICON
11540:   Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean ) : Boolean
11541:   Function OverlayIconShortCut( var Large, Small : HICON ) : Boolean
11542:   Function OverlayIconShared( var Large, Small : HICON ) : Boolean
11543:   Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList ) : string
11544:   Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11545:   Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11546:   Function ShellExecAndWait(const FileName:string;const Params:string;const Verb:string;CmdShow:Int):Bool;
11547:   Function ShellOpenAs( const FileName : string ) : Boolean

```

```

11548: Function ShellRasDial( const EntryName : string) : Boolean
11549: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11550: Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11551: TJclFileExeType, '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11552: Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11553: Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11554: Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11555: Function OemKeyScan( wOemChar : Word) : DWORD
11556: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11557: end;
11558:
11559: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11560: begin
11561: xmlVersion, 'String').SetString( '1.0 FindClass('TOBJECT'),'Exml
11562: //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11563: Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11564: Function xmlValidChar2( const Ch : WideChar) : Boolean;
11565: Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11566: Function xmlIsLetter( const Ch : WideChar) : Boolean
11567: Function xmlIsDigit( const Ch : WideChar) : Boolean
11568: Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11569: Function xmlIsNameChar( const Ch : WideChar) : Boolean
11570: Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11571: Function xmlValidName( const Text : UnicodeString) : Boolean
11572: //xmlSpace', 'Char').SetString( #$20 or #$9 or #$D or #$A);
11573: //Function xmlSkipSpace( var P : PWideChar) : Boolean
11574: //Function xmlSkipEq( var P : PWideChar) : Boolean
11575: //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11576: //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
: TUnicodeCodecClass
11577: Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11578: Function xmlTag( const Tag : UnicodeString) : UnicodeString
11579: Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11580: Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11581: Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11582: Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11583: Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11584: Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11585: Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11586: Function xmlComment( const Comment : UnicodeString) : UnicodeString
11587: Procedure SelfTestcXMLFunctions
11588: end;
11589:
11590: (*-----*)
11591: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11592: begin
11593: Function AwaitCursor : IUnknown
11594: Function ChangeCursor( NewCursor : TCursor) : IUnknown
11595: Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11596: Function YesNo( const ACaption, AMsg : string) : boolean
11597: Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11598: Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11599: Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11600: Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11601: Procedure GetSystemPaths( Strings : TStrings)
11602: Procedure MakeEditNumeric( EditHandle : integer)
11603: end;
11604:
11605: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11606: begin
11607: AddTypes('TVideoCodec', '(vcUnknown,vcRGB,vcUYU2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11608: 'BI_YUY2', 'LongWord').SetUInt( $32595559);
11609: 'BI_UYVY', 'LongWord').SetUInt( $59565955);
11610: 'BI_BTUVV', 'LongWord').SetUInt( $50313459);
11611: 'BI_YVU9', 'LongWord').SetUInt( $39555659);
11612: 'BI_YUV12', 'LongWord').SetUInt( $30323449);
11613: 'BI_Y8', 'LongWord').SetUInt( $20203859);
11614: 'BI_Y211', 'LongWord').SetUInt( $31313259);
11615: Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11616: Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11617: end;
11618:
11619: (*-----*)
11620: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11621: begin
11622: 'WM_USER', 'LongWord').SetUInt( $0400);
11623: 'WM_CAP_START', 'LongWord').SetUInt($0400);
11624: 'WM_CAP_END', 'longword').SetUInt($0400+85);
11625: //WM_CAP_START+ 85
11626: // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11627: Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11628: Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11629: Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11630: Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11631: Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11632: Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11633: Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11634: Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11635: Function capGetUserData( hwnd : THandle) : LongInt

```

```

11636: Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11637: Function capDriverDisconnect( hwnd : THandle) : LongInt
11638: Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11639: Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11640: Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11641: Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11642: Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11643: Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11644: Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11645: Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11646: Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11647: Function capEditCopy( hwnd : THandle) : LongInt
11648: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11649: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11650: Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11651: Function capDlgVideoFormat( hwnd : THandle) : LongInt
11652: Function capDlgVideoSource( hwnd : THandle) : LongInt
11653: Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11654: Function capDlgVideoCompression( hwnd : THandle) : LongInt
11655: Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11656: Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11657: Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11658: Function capPreview( hwnd : THandle; f : Word) : LongInt
11659: Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11660: Function capOverlay( hwnd : THandle; f : Word) : LongInt
11661: Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11662: Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11663: Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11664: Function capGrabFrame( hwnd : THandle) : LongInt
11665: Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11666: Function capCaptureSequence( hwnd : THandle) : LongInt
11667: Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11668: Function capCaptureStop( hwnd : THandle) : LongInt
11669: Function capCaptureAbort( hwnd : THandle) : LongInt
11670: Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11671: Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11672: Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11673: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11674: Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11675: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11676: Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11677: Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11678: Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11679: Function capPalettePaste( hwnd : THandle) : LongInt
11680: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11681: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11682: Function PCapDriverCaps, '^TCapDriverCaps // will not work
11683: TCapDriverCaps, 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11684: +; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla
11685: +; y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid
11686: +; eoIn : THandle; hVideoOut : THandle; hVideoExtIn:THandle; hVideoExtOut:THandle; end
11687: //PCapStatus, '^TCapStatus // will not work
11688: TCapStatus, 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11689: +fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT
11690: +; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO
11691: +; OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu
11692: +; rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP
11693: +; ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;
11694: +; wNumAudioAllocated : WORD; end
11695: //PCaptureParms, '^TCaptureParms // will not work
11696: TCaptureParms, 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11697: +; UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI
11698: +; ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi
11699: +; deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey
11700: +; Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl
11701: +; ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d
11702: +; wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt
11703: +; epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac
11704: +; he : BOOL; AVStreamMaster : WORD; end
11705: // PCapInfoChunk, '^TCapInfoChunk // will not work
11706: //TCapInfoChunk, 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11707: 'CONTROLCALLBACK_PREROLL', 'LongInt').SetInt( 1);
11708: 'CONTROLCALLBACK_CAPTURING', 'LongInt').SetInt( 2);
11709: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
: Integer; hwndParent : THandle; nID : Integer) : THandle
11710: Function
capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11711: 'IDS_CAP_BEGIN', 'LongInt').SetInt( 300);
11712: 'IDS_CAP_END', 'LongInt').SetInt( 301);
11713: 'IDS_CAP_INFO', 'LongInt').SetInt( 401);
11714: 'IDS_CAP_OUTOFMEM', 'LongInt').SetInt( 402);
11715: 'IDS_CAP_FILEEXISTS', 'LongInt').SetInt( 403);
11716: 'IDS_CAP_ERRORPALOPEN', 'LongInt').SetInt( 404);
11717: 'IDS_CAP_ERRORPALSAVE', 'LongInt').SetInt( 405);
11718: 'IDS_CAP_ERRORDIBSAVE', 'LongInt').SetInt( 406);
11719: 'IDS_CAP_DEFAVIEXT', 'LongInt').SetInt( 407);
11720: 'IDS_CAP_DEFAVIEXT', 'LongInt').SetInt( 408);
11721: 'IDS_CAP_CANTOPEN', 'LongInt').SetInt( 409);
11722: 'IDS_CAP_SEQ_MSGSTART', 'LongInt').SetInt( 410);

```

```

11723: 'IDS_CAP_SEQ_MSGSTOP', 'LongInt').SetInt( 411);
11724: 'IDS_CAP_VIDEDITERR', 'LongInt').SetInt( 412);
11725: 'IDS_CAP_READONLYFILE', 'LongInt').SetInt( 413);
11726: 'IDS_CAP_WRITEERROR', 'LongInt').SetInt( 414);
11727: 'IDS_CAP_NODISKSPACE', 'LongInt').SetInt( 415);
11728: 'IDS_CAP_SETFILESIZE', 'LongInt').SetInt( 416);
11729: 'IDS_CAP_SAVEASPERCENT', 'LongInt').SetInt( 417);
11730: 'IDS_CAP_DRIVER_ERROR', 'LongInt').SetInt( 418);
11731: 'IDS_CAP_WAVE_OPEN_ERROR', 'LongInt').SetInt( 419);
11732: 'IDS_CAP_WAVE_ALLOC_ERROR', 'LongInt').SetInt( 420);
11733: 'IDS_CAP_WAVE_PREPARE_ERROR', 'LongInt').SetInt( 421);
11734: 'IDS_CAP_WAVE_ADD_ERROR', 'LongInt').SetInt( 422);
11735: 'IDS_CAP_WAVE_SIZE_ERROR', 'LongInt').SetInt( 423);
11736: 'IDS_CAP_VIDEO_OPEN_ERROR', 'LongInt').SetInt( 424);
11737: 'IDS_CAP_VIDEO_ALLOC_ERROR', 'LongInt').SetInt( 425);
11738: 'IDS_CAP_VIDEO_PREPARE_ERROR', 'LongInt').SetInt( 426);
11739: 'IDS_CAP_VIDEO_ADD_ERROR', 'LongInt').SetInt( 427);
11740: 'IDS_CAP_VIDEO_SIZE_ERROR', 'LongInt').SetInt( 428);
11741: 'IDS_CAP_FILE_OPEN_ERROR', 'LongInt').SetInt( 429);
11742: 'IDS_CAP_FILE_WRITE_ERROR', 'LongInt').SetInt( 430);
11743: 'IDS_CAP_RECORDING_ERROR', 'LongInt').SetInt( 431);
11744: 'IDS_CAP_RECORDING_ERROR2', 'LongInt').SetInt( 432);
11745: 'IDS_CAP_AVI_INIT_ERROR', 'LongInt').SetInt( 433);
11746: 'IDS_CAP_NO_FRAME_CAP_ERROR', 'LongInt').SetInt( 434);
11747: 'IDS_CAP_NO_PALETTE_WARN', 'LongInt').SetInt( 435);
11748: 'IDS_CAP_MCI_CONTROL_ERROR', 'LongInt').SetInt( 436);
11749: 'IDS_CAP_MCI_CANT_STEP_ERROR', 'LongInt').SetInt( 437);
11750: 'IDS_CAP_NO_AUDIO_CAP_ERROR', 'LongInt').SetInt( 438);
11751: 'IDS_CAP_AVI_DRAWDIB_ERROR', 'LongInt').SetInt( 439);
11752: 'IDS_CAP_COMPRESSOR_ERROR', 'LongInt').SetInt( 440);
11753: 'IDS_CAP_AUDIO_DROP_ERROR', 'LongInt').SetInt( 441);
11754: 'IDS_CAP_STAT_LIVE_MODE', 'LongInt').SetInt( 500);
11755: 'IDS_CAP_STAT_OVERLAY_MODE', 'LongInt').SetInt( 501);
11756: 'IDS_CAP_STAT_CAP_INIT', 'LongInt').SetInt( 502);
11757: 'IDS_CAP_STAT_CAP_FINI', 'LongInt').SetInt( 503);
11758: 'IDS_CAP_STAT_PALETTE_BUILD', 'LongInt').SetInt( 504);
11759: 'IDS_CAP_STAT_OTPAL_BUILD', 'LongInt').SetInt( 505);
11760: 'IDS_CAP_STAT_I_FRAMES', 'LongInt').SetInt( 506);
11761: 'IDS_CAP_STAT_L_FRAMES', 'LongInt').SetInt( 507);
11762: 'IDS_CAP_STAT_CAP_L_FRAMES', 'LongInt').SetInt( 508);
11763: 'IDS_CAP_STAT_CAP_AUDIO', 'LongInt').SetInt( 509);
11764: 'IDS_CAP_STAT_VIDEOCURRENT', 'LongInt').SetInt( 510);
11765: 'IDS_CAP_STAT_VIDEOAUDIO', 'LongInt').SetInt( 511);
11766: 'IDS_CAP_STAT_VIDEOONLY', 'LongInt').SetInt( 512);
11767: 'IDS_CAP_STAT_FRAMESDROPPED', 'LongInt').SetInt( 513);
11768: 'AVICAP32', 'String').SetString( 'AVICAP32.dll
11769: end;
11770:
11771: procedure SIRegister_ALFncMisc(CL: TPSPascalCompiler);
11772: begin
11773:   function AlBoolToInt( Value : Boolean) : Integer
11774:   function AlMediumPos( LTotal, LBorder, LObject : integer) : Integer
11775:   function AlIsValidEmail( const Value : AnsiString) : boolean
11776:   function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TDateTime
11777:   function AlInc( var x : integer; Count : integer) : Integer
11778:   function AlCopyStr(const aSourceStr: AnsiString; aStart, aLength: Integer): AnsiString
11779:   function AlGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite): AnsiString;
11780:   procedure ALSaveStringToFile(Str: AnsiString; filename: AnsiString);
11781:   function ALIsInteger(const S: AnsiString): Boolean;
11782:   function ALIsDecimal(const S: AnsiString): boolean;
11783:   function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11784:   function ALWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11785:   function ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11786:   function ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11787:   function ALUTF8removeBOM(const S: AnsiString): AnsiString;
11788:   function ALRandomStr1(const aLength: Longint; const aCharSet: Array of Char): AnsiString;
11789:   function ALRandomStr(const aLength: Longint): AnsiString;
11790:   function ALRandomStrU1(const aLength: Longint; const aCharSet: Array of Char): String;
11791:   function ALRandomStrU(const aLength: Longint): String;
11792: end;
11793:
11794: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11795: begin
11796:   Procedure ALJSONToTStrings(const AJsonStr: AnsiString; aLst: TALStrings; const aNullStr: AnsiString; const
aTrueStr: AnsiString; const aFalseStr : AnsiString)
11797: end;
11798:
11799: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11800: begin
11801:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11802:   +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11803:   +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11804:   +' ; ullAvailExtendedVirtual : Int64; end
11805:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11806:   function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11807:   function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG) : LONGLONG
11808:   'INVALID_SET_FILE_POINTER', 'LongInt').SetInt( DWORD ( - 1 ));
11809:   'QUOTA_LIMITS_HARDWS_MIN_DISABLE', 'LongWord').SetUInt( $2);
11810:   'QUOTA_LIMITS_HARDWS_MIN_ENABLE', 'LongWord').SetUInt( $1);

```



```

11811: 'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11812: 'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11813: end;
11814:
11815: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11816: begin
11817:   SIRegister_THandledObject(CL);
11818:   SIRegister_TEvent(CL);
11819:   SIRegister_TMutex(CL);
11820:   SIRegister_TSharedMem(CL);
11821:   'TRACE_BUF_SIZE','LongInt').SetInt( 200 * 1024);
11822:   'TRACE_BUFFER','String').SetString( 'TRACE_BUFFER
11823:   'TRACE_MUTEX','String').SetString( 'TRACE_MUTEX
11824:   //PTraceEntry', '^TTraceEntry // will not work
11825:   SIRegister_TIPCTracer(CL);
11826:   'MAX_CLIENTS','LongInt').SetInt( 6);
11827:   'IPCTIMEOUT','LongInt').SetInt( 2000);
11828:   'IPCBUFFER_NAME','String').SetString( 'BUFFER_NAME
11829:   'BUFFER_MUTEX_NAME','String').SetString( 'BUFFER_MUTEX
11830:   'MONITOR_EVENT_NAME','String').SetString( 'MONITOR_EVENT
11831:   'CLIENT_EVENT_NAME','String').SetString( 'CLIENT_EVENT
11832:   'CONNECT_EVENT_NAME','String').SetString( 'CONNECT_EVENT
11833:   'CLIENT_DIR_NAME','String').SetString( 'CLIENT_DIRECTORY
11834:   'CLIENT_DIR_MUTEX','String').SetString( 'DIRECTORY_MUTEX
11835:   FindClass('TOBJECT'),'EMonitorActive
11836:   FindClass('TOBJECT'),'TIPCThread
11837:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna
11838:   + 'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet
11839:   + 'ach, evClientSwitch, evClientSignal, evClientExit )
11840:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11841:   TClientFlags', 'set of TClientFlag
11842:   //PEventData', '^TEventData // will not work
11843:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF
11844:   + 'lag; Flags : TClientFlags; end
11845:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11846:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11847:   TIPCTNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11848:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11849:   TIPCEventInfo', 'record FID: Integer; FKind: TEventKind; FData: TEventData; end
11850:   SIRegister_TIPCEvent(CL);
11851:   //PClientDirRecords', '^TClientDirRecords // will not work
11852:   SIRegister_TClientDirectory(CL);
11853:   TIPCState', '( stInActive, stDisconnected, stConnected )
11854:   SIRegister_TIPCThread(CL);
11855:   SIRegister_TIPCMonitor(CL);
11856:   SIRegister_TIPCCClient(CL);
11857:   Function IsMonitorRunning( var Hndl : THandle) : Boolean
11858: end;
11859:
11860: (*-----*)
11861: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11862: begin
11863:   SIRegister_TALGSMComm(CL);
11864:   Function ALGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11865:   Procedure ALGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
11866:   AMessage:AnsiString);
11867:   Function ALGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11867:   Function ALGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
11868:   UseGreekAlphabet:Bool):Widestring;
11868:   function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11869:   end;
11870:
11871: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11872: begin
11873:   TALHTTPPropertyChangeEvent', 'Procedure(sender:Tobject;const PropertyIndex:Integer;
11874:   TALHTTPProtocolVersion', '( HTTPPv_1_0, HTTPPv_1_1 )
11875:   TALHTTPMethod', '( HTTPM_Get,HTTPM_Post,HTTPM_Head,HTTPM_Trace,HTTPM_Put,HTTPM_Delete);
11876:   TInternetScheme', 'integer
11877:   TALIPv6Binary', 'array[1..16] of Char;
11878:   // TALIPv6Binary = array[1..16] of ansiChar;
11879:   // TInternetScheme = Integer;
11880:   SIRegister_TALHTTPRequestHeader(CL);
11881:   SIRegister_TALHTTPCookie(CL);
11882:   SIRegister_TALHTTPCookieCollection(CL);
11883:   SIRegister_TALHTTPResponseHeader(CL);
11884:   Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11885:   Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11886:   // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
11887:   Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11887:   // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
11888:   Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11888:   // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
11889:   Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11889:   Function ALRemoveSchemeFromUrl( aUrl : AnsiString) : ansiString
11890:   Function ALExtractSchemeFromUrl( aUrl : AnsiString) : TInternetScheme
11891:   Function ALExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11892:   Function ALExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11893:   Function ALExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11894:   Function ALInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
11894:   ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;

```

```

11895: Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password, UrlPath,
Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11896: Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11897: Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11898: Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11899: Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11900: Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11901: Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11902: Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11903: Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11904: Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11905: Function ALTryIPv4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11906: Function ALIPv4StrToNumeric( aIPv4 : ansiString) : Cardinal
11907: Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11908: Function ALZeroIPv6 : TALIPv6Binary
11909: Function ALTryIPv6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11910: Function ALIPv6StrToBinary( aIPv6 : ansiString) : TALIPv6Binary
11911: Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11912: Function ALBinaryStrToIPv6Binary( aIPv6BinaryStr : ansiString) : TALIPv6Binary
11913: end;
11914:
11915: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11916: begin
11917: Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiString;LstExtractedResourceText:TALStrings;const
DecodeHTMLText:Boolean);
11918: Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11919: Function ALXMLCDATAElementEncode( Src : AnsiString) : AnsiString
11920: Function ALXMLTextElementEncode( Src : AnsiString; const useNumericReference : boolean) : AnsiString
11921: Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
11922: Function ALUTF8HTMLEEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
useNumRef:bool):AnsiString);
11923: Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
11924: Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
11925: Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
11926: Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11927: Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
11928: end;
11929:
11930: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11931: begin
11932: SIRegister_TALEmailHeader(CL);
11933: SIRegister_TALNewsArticleHeader(CL);
11934: Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
decodeRealName:Bool):AnsiString;
11935: Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
11936: Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
11937: Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
11938: Function ALGenerateInternetMessageID : AnsiString;
11939: Function ALGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
11940: Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
11941: Function ALDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11942: end;
11943:
11944: (*-----*)
11945: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
11946: begin
11947: Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11948: Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
11949: Function ALgetLocalIPs : TALStrings
11950: Function ALgetLocalHostName : AnsiString
11951: end;
11952:
11953: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
11954: begin
11955: Procedure ALCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
TALWebRequest;ServerVariables:TALStrings);
11956: Procedure ALCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11957: Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
11958: Procedure ALCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
ScriptFileName:AnsiString;Url:AnsiString);
11959: Procedure ALCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11960: Procedure ALCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
: Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
11961: Procedure ALCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
WebRequest : TALIsapiRequest;
overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
11962: +'overloadedRequestContentStream:Tstream;var
ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
11963: Procedure ALCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
ResponseHeader : TALHTTPResponseHeader);
11964: end;
11965:
11966: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
11967: begin
11968: TStartupInfoA', 'TStartupInfo

```

```

11969: 'SE_CREATE_TOKEN_NAME', 'String').SetString( 'SeCreateTokenPrivilege
11970: SE_ASSIGNPRIMARYTOKEN_NAME', 'String').SetString( 'SeAssignPrimaryTokenPrivilege
11971: SE_LOCK_MEMORY_NAME', 'String').SetString( 'SeLockMemoryPrivilege
11972: SE_INCREASE_QUOTA_NAME', 'String').SetString( 'SeIncreaseQuotaPrivilege
11973: SE_UNSOLICITED_INPUT_NAME', 'String').SetString( 'SeUnsolicitedInputPrivilege
11974: SE_MACHINE_ACCOUNT_NAME', 'String').SetString( 'SeMachineAccountPrivilege
11975: SE_TCB_NAME', 'String').SetString( 'SeTcbPrivilege
11976: SE_SECURITY_NAME', 'String').SetString( 'SeSecurityPrivilege
11977: SE_TAKE_OWNERSHIP_NAME', 'String').SetString( 'SeTakeOwnershipPrivilege
11978: SE_LOAD_DRIVER_NAME', 'String').SetString( 'SeLoadDriverPrivilege
11979: SE_SYSTEM_PROFILE_NAME', 'String').SetString( 'SeSystemProfilePrivilege
11980: SE_SYSTEMTIME_NAME', 'String').SetString( 'SeSystemtimePrivilege
11981: SE_PROF_SINGLE_PROCESS_NAME', 'String').SetString( 'SeProfileSingleProcessPrivilege
11982: SE_INC_BASE_PRIORITY_NAME', 'String').SetString( 'SeIncreaseBasePriorityPrivilege
11983: SE_CREATE_PAGEFILE_NAME', 'String').SetString( 'SeCreatePagefilePrivilege
11984: SE_CREATE_PERMANENT_NAME', 'String').SetString( 'SeCreatePermanentPrivilege
11985: SE_BACKUP_NAME', 'String').SetString( 'SeBackupPrivilege
11986: SE_RESTORE_NAME', 'String').SetString( 'SeRestorePrivilege
11987: SE_SHUTDOWN_NAME', 'String').SetString( 'SeShutdownPrivilege
11988: SE_DEBUG_NAME', 'String').SetString( 'SeDebugPrivilege
11989: SE_AUDIT_NAME', 'String').SetString( 'SeAuditPrivilege
11990: SE_SYSTEM_ENVIRONMENT_NAME', 'String').SetString( 'SeSystemEnvironmentPrivilege
11991: SE_CHANGE_NOTIFY_NAME', 'String').SetString( 'SeChangeNotifyPrivilege
11992: SE_REMOTE_SHUTDOWN_NAME', 'String').SetString( 'SeRemoteShutdownPrivilege
11993: SE_UNDOCK_NAME', 'String').SetString( 'SeUndockPrivilege
11994: SE_SYNC_AGENT_NAME', 'String').SetString( 'SeSyncAgentPrivilege
11995: SE_ENABLE_DELEGATION_NAME', 'String').SetString( 'SeEnableDelegationPrivilege
11996: SE_MANAGE_VOLUME_NAME', 'String').SetString( 'SeManageVolumePrivilege
11997: Function ALGetEnvironmentString : AnsiString
11998: Function ALWinExec32(const FileName:CurrentDir,
Environment:AnsiString;InStream:Tstream;OutStream:Tstream):Dword;
11999: Function ALWinExec32l(const FileName:AnsiString; InputStream:Tstream;OutputStream:Tstream):Dword;
12000: Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12001: Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12002: Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12003: end;
12004:
12005: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12006: begin
12007:   Function ALEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12008:   Function ALEmptyDirectoryl( Directory : ansiString; SubDirectory : Boolean; const
RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12009:   Function ALCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12010:   Function ALGetModuleName : ansistring
12011:   Function ALGetModuleFileNameWithoutExtension : ansistring
12012:   Function ALGetModulePath : ansiString
12013:   Function ALGetFileSize( const AFileName : ansistring) : int64
12014:   Function ALGetFileVersion( const AFileName : ansistring) : ansiString
12015:   Function ALGetFileCreationDateTime( const aFileName : AnsiString) : TdateTime
12016:   Function ALGetFileLastWriteDateTime( const aFileName : AnsiString) : TdateTime
12017:   Function ALGetFileLastAccessDateTime( const aFileName : AnsiString) : TdateTime
12018:   Procedure ALSetFileCreationDateTime( const aFileName : AnsiString; const aCreationDateTime : TdateTime)
12019:   Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12020:   Function ALFileExists( const Path : ansiString) : boolean
12021:   Function ALDirectoryExists( const Directory : AnsiString) : Boolean
12022:   Function ALCreateDir( const Dir : AnsiString) : Boolean
12023:   Function ALRemoveDir( const Dir : AnsiString) : Boolean
12024:   Function ALDeleteFile( const FileName : AnsiString) : Boolean
12025:   Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12026: end;
12027:
12028: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12029: begin
12030:   NativeInt', 'Integer
12031:   NativeUInt', 'Cardinal
12032:   Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12033:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12034:   Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12035:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12036:   Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12037:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12038:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12039:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12040:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12041:   Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12042:   Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;
+ 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12044:   Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12045:   Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
OutputBuf:TByteDynArray);
12046:   Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);

```

```

12047: Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12048: Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray):NativeInt;
12049: Function ALMimeBase64DecodePartial(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12050: Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
ByteBufferSpace:Cardinal):NativeInt;
12051: Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12052: Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12053: Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12054: Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12055: Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12056: Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12057: 'cALMimeBase64_ENCODED_LINE_BREAK', 'LongInt').SetInt( 76);
12058: 'cALMimeBase64_DECODED_LINE_BREAK', 'LongInt').SetInt( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12059: 'cALMimeBase64_BUFFER_SIZE', 'LongInt').SetInt( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12060: Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12061: Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12062: Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12063: Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12064: end;
12065:
12066: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12067: begin
12068: 'cALXMLNodeMaxListSize', 'LongInt').SetInt( Maxint div 16);
12069: FindClass('TOBJECT'), 'TALXMLNode
12070: FindClass('TOBJECT'), 'TALXMLNodeList
12071: FindClass('TOBJECT'), 'TALXMLDocument
12072: TALXMLParseProcessingInstructionEvent', 'Procedure (Sender:TObject; const Target,Data:AnsiString)
12073: TALXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12074: TALXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co
12075: + 'nst Name : AnsiString; const Attributes : TALStrings)
12076: TALXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12077: TALXMLNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12078: + 'ntCDATA, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12079: + 'ntDocType, ntDocFragment, ntNotation )
12080: TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12081: TALXMLDocOptions', 'set of TALXMLDocOption
12082: TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12083: TALXMLParseOptions', 'set of TALXMLParseOption
12084: TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12085: PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12086: SIRegister_EALXMLDocError(CL);
12087: SIRegister_TALXMLNodeList(CL);
12088: SIRegister_TALXMLNode(CL);
12089: SIRegister_TALXmlElementNode(CL);
12090: SIRegister_TALXmlAttributeNode(CL);
12091: SIRegister_TALXmlTextNode(CL);
12092: SIRegister_TALXmlDocumentNode(CL);
12093: SIRegister_TALXmlCommentNode(CL);
12094: SIRegister_TALXmlProcessingInstrNode(CL);
12095: SIRegister_TALXmlCDATADataNode(CL);
12096: SIRegister_TALXmlEntityRefNode(CL);
12097: SIRegister_TALXmlEntityNode(CL);
12098: SIRegister_TALXmlDocTypeNode(CL);
12099: SIRegister_TALXmlDocFragmentNode(CL);
12100: SIRegister_TALXmlNotationNode(CL);
12101: SIRegister_TALXMLDocument(CL);
12102: cALXMLUTF8EncodingStr, 'String').SetString( 'UTF-8
12103: cALXMLUTF8HeaderStr, 'String').SetString( '<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12104: CALNSDelim, 'String').SetString( '
12105: CALXML, 'String').SetString( 'xml
12106: CALVersion, 'String').SetString( 'version
12107: CALEncoding, 'String').SetString( 'encoding
12108: CALStandalone, 'String').SetString( 'standalone
12109: CALDefaultNodeIndent, 'String').SetString( '
12110: CALXmlDocument, 'String').SetString( 'DOCUMENT
12111: Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12112: Procedure ALClearXMLDocument(const rootname:AnsiString;xmlDoc:TalXMLDocument;const
EncodingStr:AnsiString);
12113: Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12114: Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12115: Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
Recurse: Boolean):TalxmlNode
12116: Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12117: Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
AnsiString
12118: end;
12119:
12120: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12121: //based on TEEProc, TeCanvas, TEEEngine, TChart
12122: begin
12123: 'TeePiStep', 'Double').setExtended( Pi / 180.0);
12124: 'TeeDefaultPerspective', 'LongInt').SetInt( 100);
12125: 'TeeMinAngle', 'LongInt').SetInt( 270);

```



```

12126: 'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12127: 'teeclSkyBlue','LongWord').SetUInt( TColor ( $FOCAA6 ));
12128: 'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12129: 'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12130: 'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12131: 'teeclSkyBlue','LongWord').SetUInt( TColor ( $FOCAA6 ));
12132: 'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12133: 'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12134: 'TA_LEFT','LongInt').SetInt( 0);
12135: 'TA_RIGHT','LongInt').SetInt( 2);
12136: 'TA_CENTER','LongInt').SetInt( 6);
12137: 'TA_TOP','LongInt').SetInt( 0);
12138: 'TA_BOTTOM','LongInt').SetInt( 8);
12139: 'teePATCOPY','LongInt').SetInt( 0);
12140: 'NumCirclePoints','LongInt').SetInt( 64);
12141: 'teeDEFAULT_CHARSET','LongInt').SetInt( 1);
12142: 'teeANTIALIASED_QUALITY','LongInt').SetInt( 4);
12143: 'TA_LEFT','LongInt').SetInt( 0);
12144: 'bs_Solid','LongInt').SetInt( 0);
12145: 'teepf24Bit','LongInt').SetInt( 0);
12146: 'teepfDevice','LongInt').SetInt( 1);
12147: 'CM_MOUSELEAVE','LongInt').SetInt( 10000);
12148: 'CM_SYSCOLORCHANGE','LongInt').SetInt( 10001);
12149: 'DC_BRUSH','LongInt').SetInt( 18);
12150: 'DC_PEN','LongInt').SetInt( 19);
12151: teeCOLORREF', 'LongWord
12152: TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12153: //TNotifyEvent', 'Procedure ( Sender : TObject)
12154: SIRegister_TFilterRegion(CL);
12155: SIRegister_IFormCreator(CL);
12156: SIRegister_TTeeFilter(CL);
12157: //TFilterClass', 'class of TTeeFilter
12158: SIRegister_TFilterItems(CL);
12159: SIRegister_TConvolveFilter(CL);
12160: SIRegister_TBlurFilter(CL);
12161: SIRegister_TTeePicture(CL);
12162: TPenEndStyle', '( esRound, esSquare, esFlat )
12163: SIRegister_TChartPen(CL);
12164: SIRegister_TChartHiddenPen(CL);
12165: SIRegister_TDottedGrayPen(CL);
12166: SIRegister_TDarkGrayPen(CL);
12167: SIRegister_TWhitePen(CL);
12168: SIRegister_TChartBrush(CL);
12169: TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12170: TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12171: SIRegister_TView3DOptions(CL);
12172: FindClass('TOBJECT'),'TTeeCanvas
12173: TTeeTransparency', 'Integer
12174: SIRegister_TTeeBlend(CL);
12175: FindClass('TOBJECT'),'TCanvas3D
12176: SIRegister_TTeeShadow(CL);
12177: teeTGradientDirection', '( gdTopBottom, gdBottomTop, gdLeftRight, g'
12178: + 'dRightLeft, gdFromCenter, gdFromTopLeft, gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12179: FindClass('TOBJECT'),'TSubGradient
12180: SIRegister_TCustomTeeGradient(CL);
12181: SIRegister_TSubGradient(CL);
12182: SIRegister_TTeeGradient(CL);
12183: SIRegister_TTeeFontGradient(CL);
12184: SIRegister_TTeeFont(CL);
12185: TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12186: TCanvasTextAlign', 'Integer
12187: TTeeCanvasHandle', 'HDC
12188: SIRegister_TTeeCanvas(CL);
12189: TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12190: SIRegister_TFloatXYZ(CL);
12191: TPoint3D', 'record x : integer; y : integer; z : Integer; end
12192: TRGB', 'record blue: byte; green: byte; red: byte; end
12193: {TRGB=packed record
12194:   Blue : Byte;
12195:   Green : Byte;
12196:   Red : Byte;
12197:   //IFDEF CLX //Alpha : Byte; // Linux end;}
12198:
12199: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12200:   + 'TPoint3D; var Color0, Color1 : TColor) : Boolean
12201: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12202: TCanvas3DPlane', '( cpX, cpY, cpZ )
12203: //IInterface', 'IUnknown
12204: SIRegister_TCanvas3D(CL);
12205: SIRegister_TTeeCanvas3D(CL);
12206: TTrianglePoints', 'Array[0..2] of TPoint;
12207: TFourPoints', 'Array[0..3] of TPoint;
12208: Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12209: Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12210: Function Point3D( const x, y, z : Integer) : TPoint3D
12211: Procedure SwapDouble( var a, b : Double)
12212: Procedure SwapInteger( var a, b : Integer)
12213: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12214: Procedure teeRectCenter( const R : TRect; var X, Y : Integer)

```

```

12215: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12216: Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12217: Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12218: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12219: Procedure UnClipCanvas( ACanvas : TCanvas)
12220: Procedure ClipEllipse( ACanvas : TCanvas; const Rect : TRect)
12221: Procedure ClipRoundRectangle(ACanvas:TCanvas;const Rect : TRect; RoundSize : Integer)
12222: Procedure ClipPolygon(ACanvas:TCanvas;const Points:array of TPoint;NumPoints:Integer)
12223: 'TeeCharForHeight','String').SetString( 'W
12224: 'DarkerColorQuantity','Byte').SetUInt( 128);
12225: 'DarkColorQuantity','Byte').SetUInt( 64);
12226: TButtonGetColorProc', Function : TColor
12227: SIRegister_TTeeButton(CL);
12228: SIRegister_TButtonColor(CL);
12229: SIRegister_TComboFlat(CL);
12230: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TCanvasHandle)
12231: Function TeePoint( const aX, aY : Integer) : TPoint
12232: Function TeePointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12233: Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12234: Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12235: Function OrientRectangle( const R : TRect) : TRect
12236: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12237: Function PolygonBounds( const P : array of TPoint) : TRect
12238: Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12239: Function RGBValue( const Color : TColor) : TRGB
12240: Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12241: Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12242: Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12243: Function TeeCull( const P : TFourPoints) : Boolean;
12244: Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12245: TSmoothStretchOption', (' ssBestQuality, ssBestPerformance )
12246: Procedure SmoothStretch( Src, Dst : TBitmap);
12247: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12248: Function TeeDistance( const x, y : Double) : Double
12249: Function TeeLoadLibrary( const FileName : String) : HInst
12250: Procedure TeeFreeLibrary( hLibModule : HMODULE)
12251: Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12252: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12253: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12254: SIRegister_ICanvasHyperlinks(CL);
12255: SIRegister_ICanvasToolTips(CL);
12256: Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12257: end;
12258:
12259: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12260: begin
12261: TOvcHdc', 'Integer
12262: TOvcHWND', 'Cardinal
12263: TOvcHdc', 'HDC
12264: TOvcHWND', 'HWND
12265: Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12266: Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12267: Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12268: Function DefaultEpoch : Integer
12269: Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12270: Procedure FixRealPrim( P : PChar; DC : Char)
12271: Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12272: Function GetLeftButton : Byte
12273: Function GetNextDlgItem( Ctrl : TOvcHwnd) : hWnd
12274: Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12275: Function GetShiftFlags : Byte
12276: Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12277: Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12278: Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12279: Function ovIsForegroundTask : Boolean
12280: Function ovTrimLeft( const S : string) : string
12281: Function ovTrimRight( const S : string) : string
12282: Function ovQuotedStr( const S : string) : string
12283: Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12284: Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12285: Function PtrDiff( const P1, P2 : PChar) : Word
12286: Procedure PtrInc( var P, Delta : Word)
12287: Procedure PtrDec( var P, Delta : Word)
12288: Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12289: Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12290: Function ovMinI( X, Y : Integer) : Integer
12291: Function ovMaxI( X, Y : Integer) : Integer
12292: Function ovMinL( X, Y : LongInt) : LongInt
12293: Function ovMaxL( X, Y : LongInt) : LongInt
12294: Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12295: Function PartialCompare( const S1, S2 : string) : Boolean
12296: Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12297: Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12298: Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12299: Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
TransparentColor : TColor)
12300: Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
TransparentColor : TColorRef)

```

```

12301: Function ovWidthOf( const R : TRect ) : Integer
12302: Function ovHeightOf( const R : TRect ) : Integer
12303: Procedure ovDebugOutput( const S : string )
12304: Function GetArrowWidth( Width, Height : Integer ) : Integer
12305: Procedure StripCharSeq( CharSeq : string; var Str : string )
12306: Procedure StripCharFromEnd( aChr : Char; var Str : string )
12307: Procedure StripCharFromFront( aChr : Char; var Str : string )
12308: Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12309: Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12310: Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12311: Function CreateEllipticRgn( p1, p2, p3, p4 : Integer ) : HRGN
12312: Function CreateEllipticRgnIndirect( const p1 : TRect ) : HRGN
12313: Function CreateFontIndirect( const p1 : TLogFont ) : HFONT
12314: Function CreateMetaFile( p1 : PChar ) : HDC
12315: Function DescribePixelFormat( DC : HDC; p2 : Int; p3 : UINT; var p4 : TPixelFormatDescriptor ) : BOOL
12316: Function DrawText( hDC : HDC; lpString : PChar; nCount : Integer; var lpRect : TRect; uFormat : UINT ) : Integer
12317: Function DrawTextS( hDC : HDC; lpString : string; nCount : Integer; var lpRect : TRect; uFormat : UINT ) : Integer
12318: Function SetMapperFlags( DC : HDC; Flag : DWORD ) : DWORD
12319: Function SetGraphicsMode( hdc : HDC; iMode : Integer ) : Integer
12320: Function SetMapMode( DC : HDC; p2 : Integer ) : Integer
12321: Function SetMetaFileBitsEx( Size : UINT; const Data : PChar ) : HMETAFILE
12322: //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12323: Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF ) : COLORREF
12324: Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF ) : BOOL
12325: //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor ) : BOOL
12326: Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer ) : Integer
12327: Function StretchBlt( DestDC : HDC; X, Y, Width, Height : Int; SrcDC : HDC; XSrc, YSrc, SrcWidth,
SrcHeight : Int; Rop : DWORD ) : BOOL
12328: Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer ) : BOOL
12329: Function StretchDIBits( DC : HDC; DestX, DestY, DestWidth, DestHeight, SrcX, SrcY, SrcWidth,
SrcHeight : Int; Bits : int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD ) : Integer
12330: Function SetROP2( DC : HDC; p2 : Integer ) : Integer
12331: Function SetStretchBltMode( DC : HDC; StretchMode : Integer ) : Integer
12332: Function SetSystemPaletteUse( DC : HDC; p2 : UINT ) : UINT
12333: Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer ) : Integer
12334: Function SetTextColor( DC : HDC; Color : COLORREF ) : COLORREF
12335: Function SetTextAlign( DC : HDC; Flags : UINT ) : UINT
12336: Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer ) : Integer
12337: Function UpdateColors( DC : HDC ) : BOOL
12338: Function GetViewportExtEx( DC : HDC; var Size : TSize ) : BOOL
12339: Function GetViewportOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12340: Function GetWindowExtEx( DC : HDC; var Size : TSize ) : BOOL
12341: Function GetWindowOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12342: Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer ) : Integer
12343: Function InvertRgn( DC : HDC; p2 : HRGN ) : BOOL
12344: Function MaskBlt( DestDC : HDC; XDest, YDest, Width, Height : Integer; SrcDC : HDC; XSrc, YSrc : Integer;
Mask : HBITMAP; xMask, yMask : Integer; Rop : DWORD ) : BOOL
12345: Function PlgBlt( DestDC : HDC; const PtsArray, SrcDC : HDC; XSrc, YSrc, Width, Height : Int; Mask : HBITMAP; xMask,
yMask : Int ) : BOOL;
12346: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer ) : Integer
12347: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer ) : Integer
12348: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD ) : BOOL
12349: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer ) : BOOL
12350: Function PlayMetaFile( DC : HDC; MF : HMETAFILE ) : BOOL
12351: Function PaintRgn( DC : HDC; RGN : HRGN ) : BOOL
12352: Function PtInRegion( RGN : HRGN; X, Y : Integer ) : BOOL
12353: Function PtVisible( DC : HDC; X, Y : Integer ) : BOOL
12354: Function RectInRegion( RGN : HRGN; const Rect : TRect ) : BOOL
12355: Function RectVisible( DC : HDC; const Rect : TRect ) : BOOL
12356: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer ) : BOOL
12357: Function RestoreDC( DC : HDC; SavedDC : Integer ) : BOOL
12358: end;
12359:
12360: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12361: begin
12362:   SIRegister_TOvcAbstractStore(CL);
12363:   //PEXPropInfo', '^TEXPropInfo // will not work
12364:   // TEXPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12365:   SIRegister_TOvcPropertyList(CL);
12366:   SIRegister_TOvcDataFiler(CL);
12367:   Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean )
12368:   Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean )
12369:   Function CreateStoredItem( const CompName, PropName : string ) : string
12370:   Function ParseStoredItem( const Item : string; var CompName, PropName : string ) : Boolean
12371:   //Function GetPropType( PropInfo : PEXPropInfo ) : PTypeInfo
12372: end;
12373:
12374: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12375: begin
12376:   'ovsetsize', 'LongInt' ).SetInt( 16 );
12377:   'etSyntax', 'LongInt' ).SetInt( 0 );
12378:   'etSymanitic', 'LongInt' ).SetInt( 1 );
12379:   'chCR', 'Char' ).SetString( #13 );
12380:   'chLF', 'Char' ).SetString( #10 );
12381:   'chLineSeparator', '' ).SetString( chCR );
12382:   SIRegister_TCocoError(CL);
12383:   SIRegister_TCommentItem(CL);
12384:   SIRegister_TCommentList(CL);
12385:   SIRegister_TSymbolPosition(CL);

```

```

12386: TGenListType', '( glNever, glAlways, glOnError )
12387: TovBitSet', 'set of Integer
12388: //PStartTable', '^TStartTable // will not work
12389: 'TovCharSet', 'set of AnsiChar
12390: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12391: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12392: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12393: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12394: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP
12395: + 'osition; const Data : string; ErrorType : integer)
12396: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12397: TGetCH', 'Function ( pos : longint) : char
12398: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12399: SIRegister_TCocoRScanner(CL);
12400: SIRegister_TCocoRGrammar(CL);
12401: '_EF','Char').SetString( #0);
12402: '_TAB','Char').SetString( #09);
12403: '_CR','Char').SetString( #13);
12404: '_LF','Char').SetString( #10);
12405: '_EL','').SetString( _CR);
12406: '_EOF','Char').SetString( #26);
12407: 'LineEnds','TCharSet').SetInt(ord(_CR) or ord(_LF) or ord(_EF));
12408: 'minErrDist','LongInt').SetInt( 2);
12409: Function ovPadL( S : string; ch : char; L : integer) : string
12410: end;
12411:
12412: TFormatSettings = record
12413:   CurrencyFormat: Byte;
12414:   NegCurrFormat: Byte;
12415:   ThousandSeparator: Char;
12416:   DecimalSeparator: Char;
12417:   CurrencyDecimals: Byte;
12418:   DateSeparator: Char;
12419:   TimeSeparator: Char;
12420:   ListSeparator: Char;
12421:   CurrencyString: string;
12422:   ShortDateFormat: string;
12423:   LongDateFormat: string;
12424:   TimeAMString: string;
12425:   TimePMString: string;
12426:   ShortTimeFormat: string;
12427:   LongTimeFormat: string;
12428:   ShortMonthNames: array[1..12] of string;
12429:   LongMonthNames: array[1..12] of string;
12430:   ShortDayNames: array[1..7] of string;
12431:   LongDayNames: array[1..7] of string;
12432:   TwoDigitYearCenturyWindow: Word;
12433: end;
12434:
12435: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12436: begin
12437:   Function ovFormatSettings : TFormatSettings
12438: end;
12439:
12440: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12441: begin
12442:   TOvcCharSet', 'set of Char
12443:   ovBTable', 'array[0..255] of Byte
12444:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12445:   Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12446:   Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12447:   Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12448:   Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12449:   Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12450:   Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12451:   Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12452:   Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12453:   Function HexBPChar( Dest : PChar; B : Byte) : PChar
12454:   Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12455:   Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12456:   Function HexWPChar( Dest : PChar; W : Word) : PChar
12457:   Function LoCaseChar( C : Char) : Char
12458:   Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12459:   Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12460:   Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12461:   Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12462:   Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12463:   Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12464:   Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12465:   Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12466:   Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12467:   Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12468:   Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12469:   Procedure TrimAllSpacesPChar( P : PChar)
12470:   Function TrimEmbeddedZeros( const S : string) : string
12471:   Procedure TrimEmbeddedZerosPChar( P : PChar)
12472:   Function TrimTrailPrimPChar( S : PChar) : PChar
12473:   Function TrimTrailPChar( Dest, S : PChar) : PChar

```



```

12474: Function TrimTrailingZeros( const S : string) : string
12475: Procedure TrimTrailingZerosPChar( P : PChar)
12476: Function UpCaseChar( C : Char) : Char
12477: Function ovcCharInSet( C : Char; const CharSet : ToVcCharSet) : Boolean
12478: Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12479: //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12480: end;
12481:
12482: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12483: begin
12484:   //PraiseFrame', '^TRaiseFrame // will not work
12485:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : __Poin'
12486:   + 'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12487: Procedure SafeCloseHandle( var Handle : THandle)
12488: Procedure ExchangeInteger( X1, X2 : Integer)
12489: Procedure FillInteger( const Buffer, Size, Value : Integer)
12490: Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12491: Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12492:
12493: FILENAME_ADVAPI32 = 'ADVAPI32.DLL';
12494:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12495: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12496:   function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12497:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12498:   SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12499:   const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12500:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12501: function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12502:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12503:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12504:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeInfoList: PObjectTypeInfoList;
12505:   ObjectTypeInfoListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12506:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12507: function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12508:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12509:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12510:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeInfoList: PObjectTypeInfoList;
12511:   ObjectTypeInfoListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12512:   var GrantedAccess: DWORD; var AccessStatusList: DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12513: function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12514: function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12515: function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12516:   lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12517:   lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12518:   dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12519:   const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12520: function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12521: function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12522:   pSecurityDescriptor: PSecurityDescriptor; nLength: DWORD; var lpnLengthNeeded: DWORD): BOOL; stdcall;
12523: function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12524: function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12525:   dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12526: function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12527:   dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12528: function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12529:   Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12530:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12531: function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12532:   Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12533:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12534: function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12535:   lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12536: function LookupPrivilegeName(lpSystemName: PKOLChar;
12537:   var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12538: function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12539:   var lpLuid: TLargeInteger): BOOL; stdcall;
12540: function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12541:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12542: function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12543:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12544: function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12545:   ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12546:   ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12547:   var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12548:   var GenerateOnClose: BOOL): BOOL; stdcall;
12549: function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12550:   HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12551:   var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12552: function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12553: function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12554: function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12555:   ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12556: function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12557:   lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12558:   var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12559: function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12560:   var phkResult: HKEY): Longint; stdcall;
12561: function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12562:   var phkResult: HKEY): Longint; stdcall;

```

```

12563:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12564:         Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12565:         lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12566:         lpdwDisposition: PDWORD): Longint; stdcall;
12567:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12568:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12569:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12570:         var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12571:         lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12572:     function RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar; cbName: DWORD): Longint; stdcall;
12573:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12574:         var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12575:         lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12576:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12577:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY): Longint; stdcall;
12578:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12579:         ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12580:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12581:         lpcbClass: PDWORD; lpReserved: Pointer;
12582:         lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12583:         lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12584:         lpftLastWriteTime: PFileTime): Longint; stdcall;
12585:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12586:         NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotSize: DWORD): Longint; stdcall;
12587:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12588:         lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12589:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12590:         lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12591:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12592:         lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12593:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12594:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12595:         lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12596:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12597:         dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12598:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12599:         Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12600:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12601:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12602:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12603:         dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12604:         dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12605:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12606:         pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12607:
12608:     Function WAddAtom( lpString : PKOLChar ) : ATOM
12609:     Function WBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL ) : THandle
12610:     //Function WCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
12611:         lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeout : DWORD ) : BOOL
12612:     //Function WCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig ) : BOOL
12613:     Function WCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
12614:         lpString2 : PKOLChar; cchCount2 : Integer ) : Integer
12615:     Function WCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL ) : BOOL
12616:     //Function WCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
12617:         TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD ) : BOOL
12618:     Function WCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes ) : BOOL
12619:     Function WCreateDirectoryEx( lpTemplateDirectory,
12620:         lpNewDirectory: PKOLChar; lpSecAttrib: PSecurityAttribs ): BOOL;
12621:     Function WCreateEvent( lpEventAttributes: PSecurityAttrib; bManualReset,
12622:         bInitialState: BOOL; lpName: PKOLChar ): THandle;
12623:     Function WCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
12624:         PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes: DWORD; hTemplateFile: THandle ): THandle
12625:     Function WCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
12626:         dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar ) : THandle
12627:     Function WCreateHardLink( lpFileName,
12628:         lpExistingFileName: PKOLChar; lpSecurityAttributes: PSecurityAttributes ): BOOL
12629:     Function
12630:         CreateMailslot( lpName: PKOLChar; MaxMessSize: DWORD; lReadTimeout: DWORD; lpSecurityAttrib: PSecurityAttributes ): THandle;
12631:     Function WCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
12632:         nInBufferSize, nDefaultTimeout : DWORD; lpSecurityAttributes : PSecurityAttributes ) : THandle
12633:     //Function WCreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
12634:         lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
12635:         Pointer; lpCurrentDirectory: PKOLChar; const lpStartupInfo: TStartupInfo; var
12636:         lpProcessInfo: TProcessInformation ): BOOL
12637:     Function WCreateSemaphore( lpSemaphoreAttributes : PSecurityAttributes; lInitialCount, lMaximumCount :
12638:         Longint; lpName : PKOLChar ) : THandle
12639:     Function
12640:         WCreateWaitableTimer( lpTimerAttributes: PSecurityAttribs; bManualReset: BOOL; lpTimerName: PKOLChar ): THandle;
12641:     Function WDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar ) : BOOL
12642:     Function WDeleteFile( lpFileName : PKOLChar ) : BOOL
12643:     Function WEndUpdateResource( hUpdate : THandle; fDiscard : BOOL ) : BOOL
12644:     //Function
12645:         WEnumCalendarInfo( lpCalInfEnumProc: TFNCalInfEnumProc; Locale: LCID; Calendar: CALID; CalType: CALTYPE ): BOOL;
12646:     //Function WEnumDateFormats( lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD ) : BOOL
12647:     //Function WEnumResourceLanguages( hModule: HMODULE; lpType,
12648:         lpName: PChar; lpEnumFunc: ENUMRESLANGPROC; lParam: Longint : BOOL )
12649:     //Function
12650:         WEnumResourceNames( hModule: HMODULE; lpType: PKOLChar; lpEnumFunc: ENUMRESNAMEPROC; lParam: Longint ): BOOL;
12651:     //Function WEnumResourceTypes( hModule: HMODULE; lpEnumFunc: ENUMRESTYPEPROC; lParam: Longint ): BOOL;

```

```

12634: //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCCodepageEnumProc; dwFlags : DWORD) : BOOL
12635: //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12636: //Function wEnumTimeFormats( lpTimeFmtEnumProc : TFNTTimeFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL;
12637: Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12638: Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12639: //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12640: Function wFindAtom( lpString : PKOLChar) : ATOM
12641: Function
wFindFirstChangeNotification( lpPathName : PKOLChar; bWatchSubtree : BOOL; dwNotifyFilter : DWORD) : THandle;
12642: Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12643: //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindInfoLevels; lpFindFileData :
Pointer; fSearchOp : TFindSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12644: Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12645: Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12646: Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12647: Function
wFoldString( dwMapFlags : DWORD; lpSrcStr : PKOLChar; cchSrc : Int; lpDestStr : PKOLChar; cchDest : Integer) : Integer;
12648: //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12649: Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12650: Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12651: Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12652: Function wGetCommandLine : PKOLChar
12653: //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12654: Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12655: Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12656: //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12657: Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12658: //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
lpDateStr : PKOLChar; cchDate : Integer) : Integer
12659: //Function wGetDefaultCommConfig( lpszName : PKOLChar; var lpCC : TCommConfig; var lpdwSize : DWORD) : BOOL
12660: Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12661: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12662: Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12663: Function wGetEnvironmentStrings : PKOLChar
12664: Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12665: Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12666: //Function
wGetFileAttributesEx( lpFileName : PKOLChar; fInfoLevelId : TGetFileExInfoLevs; lpFileInform : Pointer) : BOOL;
12667: Function wGetFullPathName( lpFileName : PKOLChar; nBufferLength : WORD; lpBuffer : PKOLChar; var
lpFilePart : PKOLChar) : DWORD;
12668: //Function wGetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCData : PKOLChar; cchData : Integer) : Integer
12669: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12670: Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12671: Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12672: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12673: //Function wGetNumberFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat : PNumberFmt;
lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12674: Function wGetPrivateProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer; lpFileName : PKOLChar) : UINT;
12675: Function
wGetPrivateProfileSection( lpAppName : PKOLChar; lpRetrStr : PKOLChar; nSize : DWORD; pFileName : PKOLChar) : DWORD;
12676: Function wGetPrivateProfileSectionNames( lpszReturnBuffer : PKOLChar; nSize : DWORD; lpFileName : PKOLChar) : DWORD;
12677: Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar; lpReturnedStr : PKOLChar;
nSize : DWORD; lpFileName : PKOLChar) : DWORD
12678: Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12679: Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12680: Function wGetProfileString( lpAppName, lpKeyName,
lpDefault : PKOLChar; lpReturnedStr : PKOLChar; nSize : DWORD) : DWORD;
12681: Function wGetShortPathName( lpszLongPath : PKOLChar; lpszShortPath : PKOLChar; cchBuffer : DWORD) : DWORD
12682: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12683: //Function wGetStringTypeEx( Locale : LCID; dwInfoType : DWORD; lpSrcStr : PKOLChar; cchSrc : Integer; var
lpCharType) : BOOL
12684: Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12685: Function wGetTempFileName( lpPathName, lpPrefixString : PKOLChar; uUnique : UINT; lpTempFileName : PKOLChar) : UINT
12686: Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12687: //Function
wGetTimeFormat( Loc : LCID; dwFlgs : DWORD; lpTime : PSystemTime; lpFrm : PKOLChar; lpTimeStr : PKOLChar; cTime : Int) : Int
12688: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12689: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize :
DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12690: Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12691: Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12692: Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12693: Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12694: Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12695: Function
wLcMapString( Loc : LCID; dwMapFlgs : DWORD; lpSrcStr : PKOLChar; cchSrc : Int; lpDestStr : PKOLChar; cchDest : Int) : Int;
12696: Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12697: Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12698: Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12699: Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12700: //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL

```



```

12701: Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName:PKOLChar ) : THandle
12702: Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12703: Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12704: Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12705: Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12706: Procedure wOutputDebugString( lpOutputString : PKOLChar)
12707: //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
lpNumberOfEventsRead:DWORD):BOOL;
12708: Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12709: //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12710: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12711: //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
lpNumbOfEventsRead:DWORD):BOOL;
12712: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
: TCoord; var lpReadRegion : TSmallRect) : BOOL
12713: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12714: Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12715: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12716: Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12717: Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12718: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12719: Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12720: //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12721: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12722: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12723: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCData : PKOLChar) : BOOL
12724: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12725: //Function wUpdateResource(hUpdate:THandle;lpType,
lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12726: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12727: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12728: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12729: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
var lpNumberOfEventsWritten : DWORD) : BOOL
12730: //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
TCoord; var lpWriteRegion : TSmallRect) : BOOL
12731: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12732: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12733: Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12734: Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12735: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12736: Function wlstcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12737: Function wlstncmp( lpString1, lpString2 : PKOLChar) : Integer
12738: Function wlstncmpi( lpString1, lpString2 : PKOLChar) : Integer
12739: Function wlstncpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12740: Function wlstncpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12741: Function wlstrlen( lpString : PKOLChar) : Integer
12742: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
PNetConnectInfoStruct) : DWORD
12743: //Function wNetAddConnection2(var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12744: //Function wNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12745: Function wNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12746: Function wNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12747: Function wNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12748: //Function wNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12749: //Function wNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12750: Function wNetEnumResource(hEnum:THandle;var lpCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12751: Function wNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12752: Function wNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
: PKOLChar; nNameBufSize : DWORD) : DWORD
12753: //Function wNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12754: Function wNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12755: //Function wNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12756: //Function wNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
lpBufferSize:DWORD):DWORD;
12757: Function wNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12758: //Function wNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lpEnum:THandle):DWORD;
12759: //Function wNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12760: //Function wNetUseConnection(hwndOwner:HWND;var
lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
lpBufferSize:DWORD;var lpResult:DWORD) : DWORD
12761: Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12762: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12763: Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12764: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12765: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12766: //Function wGetPrivateProfileStruct(lpszSection,
lpSzKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12767: //Function wWritePrivateProfileStruct(lpszSection,
lpSzKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;

```



```

12768: Function wAddFontResource( FileName : PKOLChar ) : Integer
12769: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : Integer
12770: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar ) : HENHMETAFILE
12771: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar ) : HMETAFILE
12772: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace ) : HCOLORSPACE
12773: //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvInit : PDeviceMode ) : HDC
12774: //Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar ) : HDC
12775: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientation, fnWeight : Integer; fdwItalic,
fdwUnderline, fdwStrikeOut, fdwCharSet, fdwOutputPrec, fdwClipPrecision, fdwQualy,
fdwPitchAndFamily : DWORD; lpszFace : PKOLChar ) : HFONT;
12776: Function wCreateFontIndirect( const p1 : TLogFont ) : HFONT
12777: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV ) : HFONT
12778: //Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvInit : PDeviceMode ) : HDC
12779: Function wCreateMetaFile( p1 : PKOLChar ) : HDC
12780: Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar ) : BOOL
12781: //Function wDeviceCapabilities( pDriverName, pDeviceName,
pPort : PKOLChar; iIdx : Integer; pOut : PKOLChar; DevMod : PDeviceMode ) : Integer;
12782: //Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFontEnumProc; p4 : LPARAM ) : BOOL
12783: //Function wEnumFontFamiliesEx( DC : HDC; var p2 : TLogFont; p3 : TFontEnumProc; p4 : LPARAM; p5 : DWORD ) : BOOL;
12784: //Function wEnumFonts( DC : HDC; lpszFace : PKOLChar; fntempProc : TFontEnumProc; lpszData : PKOLChar ) : Integer;
12785: //Function wEnumICMProfiles( DC : HDC; ICMProc : TFontEnumProc; p3 : LPARAM ) : Integer
12786: //Function wExtTextOut( DC : HDC; X,
Y : Integer; Options : Longint; Rect : PRect; Str : PKOLChar; Count : Longint; Dx : Integer ) : BOOL
12787: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : Integer; const ABCStructs ) : BOOL
12788: //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : Integer; const ABCFloatStructs ) : BOOL
12789: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : Integer; const Widths ) : BOOL
12790: //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : Integer; const Widths ) : BOOL
12791: //Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : Integer; const Widths ) : BOOL
12792: //Function wGetCharacterPlacement( DC : HDC; p2 : PKOLChar; p3, p4 : Integer; var p5 : TGCPRResults; p6 : DWORD ) : DWORD
12793: Function wGetEnhMetaFile( p1 : PKOLChar ) : HENHMETAFILE
12794: Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : Integer; p3 : PKOLChar ) : Integer
12795: //Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD ) : DWORD
12796: //Function wGetGlyphOutline( DC : HDC; uChar, uFormat : Integer; const lpgm : TGlyphMetrics; cbBuffer : DWORD;
lpvBuffer : Pointer; const lpmat2 : TMat2 ) : DWORD
12797: Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar ) : BOOL
12798: //Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD ) : BOOL
12799: Function wGetMetaFile( p1 : PKOLChar ) : HMETAFILE
12800: //Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer ) : Integer
12801: //Function wGetOutlineTextMetrics( DC : HDC; p2 : Integer; OTMetricStructs : Pointer ) : Integer
12802: //Function wGetTextExtentExPoint( DC : HDC; p2 : PKOLChar; p3, p4 : Integer; p5, p6 : Integer; var p7 : TSize ) : BOOL
12803: Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize ) : BOOL
12804: Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize ) : BOOL
12805: Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar ) : Integer
12806: //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric ) : BOOL
12807: Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer ) : BOOL
12808: Function wRemoveFontResource( FileName : PKOLChar ) : BOOL
12809: //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : BOOL
12810: //Function wResetDC( DC : HDC; const InitData : TDeviceMode ) : HDC
12811: Function wSetICMProfile( DC : HDC; Name : PKOLChar ) : BOOL
12812: //Function wStartDoc( DC : HDC; const p2 : TDocInfo ) : Integer
12813: Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer ) : BOOL
12814: Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : Integer ) : BOOL
12815: Function wglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD ) : BOOL
12816: //Function wglUseFontOutlines( p1 : HDC; p2, p3, p4 : DWORD; p5, p6 : Single; p7 : Integer; p8 : PGlyphMetricsFloat ) : BOOL
12817: Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : Integer; lpNewItem : PKOLChar ) : BOOL
12818: Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer ) : BOOL
12819: //Function
wCallWindowProc( lpPrevWndFunc : TFontEnumProc; hWnd : HWND; Msg : Integer; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12820: //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD ) : Longint
12821: //Function wChangeDisplaySettingsEx( lpszDeviceName : PKOLChar; var lpDevMode : TDeviceMode; hWnd : HWND;
dwFlags : DWORD; lParam : Pointer ) : Longint
12822: Function wChangeMenu( hMenu : HMENU; cmd : Integer; lpszNewItem : PKOLChar; cmdInsert : Integer; flags : Integer ) : BOOL;
12823: Function wCharLower( lpsz : PKOLChar ) : PKOLChar
12824: Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12825: Function wCharNext( lpsz : PKOLChar ) : PKOLChar
12826: //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12827: Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar ) : PKOLChar
12828: //Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12829: Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12830: Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12831: Function wCharUpper( lpsz : PKOLChar ) : PKOLChar
12832: Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12833: Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer ) : Integer
12834: Function wCreateAcceleratorTable( var Accel, Count : Integer ) : HACCEL
12835: //Function wCreateDesktop( lpszDesktop,
lpszDevice : PKOLChar; pDevMode : PDeviceMode; dwFlags : DWORD; dwDesiredAccess : DWORD; lpsa : PSecurityAttribs ) : HDESK
12836: //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
HWND; lpDialogFunc : TFontEnumProc; dwInitParam : LPARAM ) : HWND
12837: //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
lpDialogFunc : TFontEnumProc; dwInitParam : LPARAM ) : HWND
12838: Function wCreateMDIWindow( lpClassName, lpWindowName : PKOLChar; dwStyle : DWORD; X, Y, nWidth, nHeight : Integer;
hWndParent : HWND; hInstance : HINST; lParam : LPARAM ) : HWND
12839: //Function wCreateWindowEx( dwExStyle : DWORD; lpClassName : PKOLChar; lpWindowName : PKOLChar; dwStyle : DWORD; X, Y,
nWidth, nHeight : Integer; hWndParent : HWND; hMenu : HMENU; hInstance : HINST; lParam : Pointer ) : HWND
12840: //Function wCreateWindowStation( lpwinsta : PKOLChar; dwReserv,
dwDesiredAccess : DWORD; lpsa : PSecurityAttribs ) : HWINSTA;
12841: Function wDefDlgProc( hWnd : HWND; Msg : Integer; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12842: Function wDefFrameProc( hWnd, hWndMDIClient : HWND; uMsg : Integer; wParam : WPARAM; lParam : LPARAM ) : LRESULT;
12843: Function wDefMDIChildProc( hWnd : HWND; uMsg : Integer; wParam : WPARAM; lParam : LPARAM ) : LRESULT;

```

```

12844: Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12845: //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
: HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12846: //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
: TFNDlgProc; dwInitParam : LPARAM ) : Integer
12847: Function wDispatchMessage( const lpMsg : TMsg ) : Longint
12848: Function wDlglDirList( hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
nIDStaticPath:Integer;uFileType:UINT):Integer;
12849: Function wDlglDirListComboBox( hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
nIDStaticPath:Integer;uFileType:UINT):Integer;
12850: Function wDlglDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer) : BOOL
12851: Function wDlglDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer ) : BOOL
12852: //Function wDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
cy:Integer;Flags:UINT):BOOL;
12853: Function wDrawText( hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12854: Function wFindWindow( lpClassName, lpWindowName : PKOLChar ) : HWND
12855: Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar ) : HWND
12856: //Function wGetAltTabInfo(hwnd:HWND;iItem:Integer;var
pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12857: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass ) : BOOL
12858: //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx ) : BOOL
12859: Function wGetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD
12860: Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer ) : Integer
12861: Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12862: Function wGetDlglItemText( hDlg:HWND;nIDlglItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12863: Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
12864: Function wGetKeyboardLayoutName( pwszKLID : PKOLChar ) : BOOL
12865: //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo ) : BOOL
12866: Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12867: Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wParam: WPARAM; lParam: LPARAM ) : Integer
12868: Function wGetProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12869: //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
lpnTabStopPositions ) : DWORD
12870: //Function wGetObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Ptr;nLength:DWORD;var
lpnLengthNeed:DWORD)BOOL;
12871: Function wGetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
12872: Function wGetWindowModuleFileName( hWnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT ) : UINT
12873: Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer ) : Integer
12874: Function wGetWindowTextLength( hWnd : HWND ) : Integer
12875: //Function wGrayString( hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
nHeigt:Integer):BOOL;
12876: Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12877: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12878: Function wIsCharAlpha( ch : KOLChar ) : BOOL
12879: Function wIsCharAlphaNumeric( ch : KOLChar ) : BOOL
12880: Function wIsCharLower( ch : KOLChar ) : BOOL
12881: Function wIsCharUpper( ch : KOLChar ) : BOOL
12882: Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg ) : BOOL
12883: Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar ) : HACCEL
12884: Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar ) : HBITMAP
12885: Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar ) : HCURSOR
12886: Function wLoadCursorFromFile( lpFileName : PKOLChar ) : HCURSOR
12887: Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar ) : HICON
12888: Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12889: Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
12890: Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar ) : HMENU
12891: //Function wLoadMenuIndirect( lpMenuTemplate : Pointer ) : HMENU
12892: Function wLoadString(hInstance:HINST;uID:UINT;lpBuffer:PKOLChar;nBufferMax:Integer):Integer
12893: Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
12894: Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwHKL : HKL ) : UINT
12895: Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
12896: Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12897: //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL
12898: Function wModifyMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12899: //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR ) : BOOL
12900: //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD ) : BOOL
12901: //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12902: Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12903: Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12904: Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD ) : HWINSTA
12905: Function wPeekMessage( var lpMsg : TMsg; hWnd : HWND; wParam: WPARAM; lParam: LPARAM ) : Integer
12906: Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12907: Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12908: Function wRealGetWindowClass( hWnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
12909: // Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
12910: // Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM
12911: Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
12912: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12913: Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
12914: Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12915: Function wSendDlglItemMessage(hDlg:HWND;nIDlglItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12916: Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12917: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
lpResultCallBack : TFNSendAsyncProc; dwData : DWORD ) : BOOL
12918: Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM;fuFlags,uTimeout:UINT;var
lpdwResult:DWORD): LRESULT
12919: Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12920: Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
12921: Function wSetDlglItemText( hDlg : HWND; nIDlglItem : Integer; lpString : PKOLChar ) : BOOL

```

```

12922: //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12923: Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
12924: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
12925: Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
12926: Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
12927: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
12928: //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12929: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
12930: Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
  lpnTabStopPositions,nTabOrigin:Int):Longint;
12931: Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
12932: Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
12933: Function wVkKeyScan( ch : KOLChar) : SHORT
12934: Function wVkKeyScanEx( ch : KOLChar; dwHKL : HKL) : SHORT
12935: Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
12936: Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
12937: Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
12938:
12939: //TestDrive!
12940: 'SID_REVISION', 'LongInt').SetInt(1); 'FILENAME_ADVAPI32', 'String').SetString('ADVAPI32.DLL
12941: 'PROC_CONVERTSIDTOSTRINGSIDA', 'String').SetString('ConvertSidToStringSidA
12942: Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
12943: Function GetLocalUserSidStr( const UserName : string) : string
12944: Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
12945: Function Impersonate2User( const domain : string; const user : string) : boolean
12946: Function GetProcessUserByPid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
12947: Function KillProcessbyname( const exename : string; var found : integer) : integer
12948: Function getWinProcessList : TStringList
12949: end;
12950:
12951: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
12952: begin
12953:   'AfMaxSyncSlots', 'LongInt').SetInt( 64);
12954:   'AfSynchronizeTimeout', 'LongInt').SetInt( 2000);
12955:   'TafSyncSlotID', 'DWORD
12956:   'TafSyncStatistics', 'record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
12957:   'TafSafeSyncEvent', 'Procedure ( ID : TafSyncSlotID)
12958:   'TafSafeDirectSyncEvent', 'Procedure
12959:   Function AfNewSyncSlot( const AEvent : TafSafeSyncEvent) : TafSyncSlotID
12960:   Function AfReleaseSyncSlot( const ID : TafSyncSlotID) : Boolean
12961:   Function AfEnableSyncSlot( const ID : TafSyncSlotID; Enable : Boolean) : Boolean
12962:   Function AfValidateSyncSlot( const ID : TafSyncSlotID) : Boolean
12963:   Function AfSyncEvent( const ID : TafSyncSlotID; Timeout : DWORD) : Boolean
12964:   Function AfDirectSyncEvent( Event : TafSafeDirectSyncEvent; Timeout : DWORD) : Boolean
12965:   Function AfIsSyncMethod : Boolean
12966:   Function AfSyncWnd : HWND
12967:   Function AfSyncStatistics : TafSyncStatistics
12968:   Procedure AfClearSyncStatistics
12969: end;
12970:
12971: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
12972: begin
12973:   'fBinary', 'LongWord').SetUInt( $00000001);
12974:   'fParity', 'LongWord').SetUInt( $00000002);
12975:   'fOutxCtsFlow', 'LongWord').SetUInt( $00000004);
12976:   'fOutxDsrFlow', 'LongWord').SetUInt( $00000008);
12977:   'fDtrControl', 'LongWord').SetUInt( $00000030);
12978:   'fDtrControlDisable', 'LongWord').SetUInt( $00000000);
12979:   'fDtrControlEnable', 'LongWord').SetUInt( $00000010);
12980:   'fDtrControlHandshake', 'LongWord').SetUInt( $00000020);
12981:   'fDsrSensitivity', 'LongWord').SetUInt( $00000040);
12982:   'fTXContinueOnXoff', 'LongWord').SetUInt( $00000080);
12983:   'fOutX', 'LongWord').SetUInt( $00000100);
12984:   'fInX', 'LongWord').SetUInt( $00000200);
12985:   'fErrorChar', 'LongWord').SetUInt( $00000400);
12986:   'fNull', 'LongWord').SetUInt( $00000800);
12987:   'fRtsControl', 'LongWord').SetUInt( $00003000);
12988:   'fRtsControlDisable', 'LongWord').SetUInt( $00000000);
12989:   'fRtsControlEnable', 'LongWord').SetUInt( $00001000);
12990:   'fRtsControlHandshake', 'LongWord').SetUInt( $00002000);
12991:   'fRtsControlToggle', 'LongWord').SetUInt( $00003000);
12992:   'fAbortOnError', 'LongWord').SetUInt( $00004000);
12993:   'fDummy2', 'LongWord').SetUInt( $FFFF8000);
12994:   'TafCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
12995:   FindClass('TOBJECT'), 'EAfComPortCoreError
12996:   FindClass('TOBJECT'), 'TafComPortCore
12997:   'TafComPortCoreEvent', 'Procedure ( Sender : TafComPortCore; Even'
12998:   + 'tKind : TafCoreEvent; Data : DWORD)
12999:   SIRegister_TafComPortCoreThread(CL);
13000:   SIRegister_TafComPortEventThread(CL);
13001:   SIRegister_TafComPortWriteThread(CL);
13002:   SIRegister_TafComPortCore(CL);
13003:   Function FormatDeviceName( PortNumber : Integer) : string
13004: end;
13005:
13006: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13007: begin
13008:   'TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13009:   'TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean

```

```

13010: SIRegister_TApplicationFileIO(CL);
13011: TDataFileCapability', '( dfcRead, dfcWrite )
13012: TDataFileCapabilities', 'set of TDataFileCapability
13013: SIRegister_TDataFile(CL);
13014: //TDataFileClass', 'class of TDataFile
13015: Function ApplicationFileIODefined : Boolean
13016: Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13017: Function FileStreamExists(const fileName: String) : Boolean
13018: //Procedure Register
13019: end;
13020:
13021: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13022: begin
13023:   TALFBXFieldType', '( uftUnknown, uftNumeric, uftChar, uftVarchar'
13024:   +', uftCString, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13025:   +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13026:   TALFBXScale', 'Integer
13027:   FindClass('TOBJECT'),'EALFBXConvertError
13028:   SIRegister_EALFBXError(CL);
13029:   SIRegister_EALFBXException(CL);
13030:   FindClass('TOBJECT'),'EALFBXGfixError
13031:   FindClass('TOBJECT'),'EALFBXDSQLError
13032:   FindClass('TOBJECT'),'EALFBXDynError
13033:   FindClass('TOBJECT'),'EALFBXGBakError
13034:   FindClass('TOBJECT'),'EALFBXGSecError
13035:   FindClass('TOBJECT'),'EALFBXLicenseError
13036:   FindClass('TOBJECT'),'EALFBXGStatError
13037:   //EALFBXExceptionClass', 'class of EALFBXError
13038:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13039:   + '37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13040:   + 'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13041:   + 'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252, '
13042:   + 'csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13043:   + 'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13044:   + '_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13045:   + '8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13046:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13047:   + 'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13048:   + 'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13049:   + 'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13050:   TALFBXTransParams', 'set of TALFBXTransParam
13051:   Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13052:   Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13053:   Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13054:   'cALFBXMaxParamLength', 'LongInt').SetInt( 125);
13055:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13056:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13057:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13058:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13059:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete, '
13060:   + 'stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13061:   + 't, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13062:   SIRegister_TALFBXSQLDA(CL);
13063:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13064:   SIRegister_TALFBXPoolStream(CL);
13065:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13066:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13067:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13068:   //TALFBXArrayDesc', 'TISCArrDesc
13069:   //TALFBXBlobDesc', 'TISCBlobDesc
13070:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13071:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13072:   SIRegister_TALFBXSQLResult(CL);
13073:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13074:   SIRegister_TALFBXSQLParams(CL);
13075:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13076:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13077:   + 'atementType : TALFBXStatementType; end
13078:   FindClass('TOBJECT'),'TALFBXLibrary
13079:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13080:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13081:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13082:   //+ ' Excep : EALFBXExceptionClass)
13083:   SIRegister_TALFBXLibrary(CL);
13084:   'cALFBXDateOffset', 'LongInt').SetInt( 15018);
13085:   'cALFBXTimeCoeff', 'LongInt').SetInt( 864000000);
13086:   //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13087:   //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13088:   //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13089:   Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13090:   Procedure ALFBXDecodeSQLTime(v: Cardinal; out Hour, Minute, Second: Word; out Fractions: LongWord)
13091:   //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13092:   //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13093:   //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13094:   Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13095:   Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13096:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13097:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13098:   Function ALFBXSQLQuote( const name : AnsiString) : AnsiString

```



```

13099: Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13100: end;
13101:
13102: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13103: begin
13104:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13105:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13106:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13107:   + 'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13108:   + 'teger; First : Integer; CacheThreshold : Integer; end
13109:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13110:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13111:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13112:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13113:   + '_writes : int64; page_fetches : int64; page_marks : int64; end
13114:   SIRegister_TALFBXClient(CL);
13115:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13116:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13117:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13118:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13119:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13120:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13121:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13122:   SIRegister_TALFBXConnectionPoolClient(CL);
13123:   SIRegister_TALFBXEventThread(CL);
13124:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13125: end;
13126:
13127: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13128: begin
13129:   _OSVERSIONINFOA = record
13130:     dwOSVersionInfoSize: DWORD;
13131:     dwMajorVersion: DWORD;
13132:     dwMinorVersion: DWORD;
13133:     dwBuildNumber: DWORD;
13134:     dwPlatformId: DWORD;
13135:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13136:   end;
13137:   TOSVersionInfoA', '_OSVERSIONINFOA
13138:   TOSVersionInfo', 'TOSVersionInfoA
13139:   'WS_EX_RIGHT', 'LongWord').SetUInt( $00001000);
13140:   'WS_EX_LEFT', 'LongWord').SetUInt( $00000000);
13141:   'WS_EX_RTLREADING', 'LongWord').SetUInt( $00002000);
13142:   'WS_EX_LTRREADING', 'LongWord').SetUInt( $00000000);
13143:   'WS_EX_LEFTSCROLLBAR', 'LongWord').SetUInt( $00004000);
13144:   'WS_EX_RIGHTSCROLLBAR', 'LongWord').SetUInt( $00000000);
13145:   Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13146:   'LAYOUT_RTL', 'LongWord').SetUInt( $00000001);
13147:   'LAYOUT_BTT', 'LongWord').SetUInt( $00000002);
13148:   'LAYOUT_VBH', 'LongWord').SetUInt( $00000004);
13149:   'LAYOUT_BITMAPORIENTATIONPRESERVED', 'LongWord').SetUInt( $00000008);
13150:   'NOMIRRORBITMAP', 'LongWord').SetUInt( DWORD ( $80000000 ));
13151:   Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13152: Function GetLayout( dc : hdc) : DWORD
13153: Function IsBidi : Boolean
13154: Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13155: Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13156: Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13157: Function GetPriorityClass( hProcess : THandle) : DWORD
13158: Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13159: Function CloseClipboard : BOOL
13160: Function GetClipboardSequenceNumber : DWORD
13161: Function GetClipboardOwner : HWND
13162: Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13163: Function GetClipboardViewer : HWND
13164: Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13165: Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13166: Function GetClipboardData( uFormat : UINT) : THandle
13167: Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13168: Function CountClipboardFormats : Integer
13169: Function EnumClipboardFormats( format : UINT) : UINT
13170: Function GetClipboardFormatName( format: UINT; lpszFormatName: PChar; cchMaxCount: Integer): Integer
13171: Function EmptyClipboard : BOOL
13172: Function IsClipboardFormatAvailable( format : UINT) : BOOL
13173:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13174: Function GetOpenClipboardWindow : HWND
13175: Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13176: Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13177: Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13178: Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13179: Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13180: Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13181: Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton, nIDCheckButton : Integer) : BOOL
13182: Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13183: Function SendDlgItemMessage( hDlg:HWND;nIDDlgItem: Int;Msg: UINT;wParam: WPARAM;lParam: LPARAM): Longint;
13184: end;
13185:
13186: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13187: begin

```

```

13188: Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13189: Function GetTemporaryFilesPath : String
13190: Function GetTemporaryFileName : String
13191: Function FindFileInPaths( const fileName, paths : String ) : String
13192: Function PathsToString( const paths : TStrings) : String
13193: Procedure StringToPaths( const pathsString : String; paths : TStrings)
13194: //Function MacroExpandPath( const aPath : String) : String
13195: end;
13196:
13197: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13198: begin
13199:   SIRegister_TALMultiPartBaseContent(CL);
13200:   SIRegister_TALMultiPartBaseContents(CL);
13201:   SIRegister_TALMultiPartBaseStream(CL);
13202:   SIRegister_TALMultiPartBaseEncoder(CL);
13203:   SIRegister_TALMultiPartBaseDecoder(CL);
13204: Function ALMultiPartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13205: Function ALMultiPartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13206: Function ALMultiPartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13207: end;
13208:
13209: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13210: begin
13211:   TDriveSize', 'record FreeS : Int64; Totals : Int64; end
13212:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : Int'
13213:   +eger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13214: Function aAllocPadedMem( Size : Cardinal) : TObjct
13215: Procedure aFreePadedMem( var P : TObjct);
13216: Procedure aFreePadedMeml( var P : PChar);
13217: Function aCheckPadedMem( P : Pointer) : Byte
13218: Function aGetPadMemSize( P : Pointer) : Cardinal
13219: Function aAllocMem( Size : Cardinal) : Pointer
13220: Function aStrLen( const Str : PChar) : Cardinal
13221: Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13222: Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13223: Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13224: Function aStrEnd( const Str : PChar) : PChar
13225: Function aStrScan( const Str : PChar; aChr : Char) : PChar
13226: Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13227: Function aPCharLength( const Str : PChar) : Cardinal
13228: Function aPCharUpper( Str : PChar) : PChar
13229: Function aPCharLower( Str : PChar) : PChar
13230: Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13231: Function aLastDelimiter( const Delimiters, S : String) : Integer
13232: Function aCopyTail( const S : String; Len : Integer) : String
13233: Function aInt2Thos( I : Int64) : String
13234: Function aUpperCase( const S : String) : String
13235: Function aLowerCase( const S : string) : String
13236: Function aCompareText( const S1, S2 : string) : Integer
13237: Function aSameText( const S1, S2 : string) : Boolean
13238: Function aInt2Str( Value : Int64) : String
13239: Function aStr2Int( const Value : String) : Int64
13240: Function aStr2IntDef( const S : string; Default : Int64) : Int64
13241: Function aGetFileExt( const FileName : String) : String
13242: Function aGetFilePath( const FileName : String) : String
13243: Function aGetFileName( const FileName : String) : String
13244: Function aChangeExt( const FileName, Extension : String) : String
13245: Function aAdjustLineBreaks( const S : string) : string
13246: Function aGetWindowStr( WinHandle : HWND) : String
13247: Function aDiskSpace( Drive : String) : TDriveSize
13248: Function aFileExists( FileName : String) : Boolean
13249: Function aFileSize( FileName : String) : Int64
13250: Function aDirectoryExists( const Name : string) : Boolean
13251: Function aSysErrorMessage( ErrorCode : Integer) : string
13252: Function aShortPathName( const LongName : string) : string
13253: Function aGetWindowVer : TWinVerRec
13254: procedure InitDriveSpacePtr;
13255: end;
13256:
13257: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13258: begin
13259:   aZero', 'LongInt').SetInt( 0);
13260:   'makeappDEF', 'LongInt').SetInt( - 1);
13261:   'CS_VREDRAW', 'LongInt').SetInt( DWORD ( 1 ));
13262:   'CS_HREDRAW', 'LongInt').SetInt( DWORD ( 2 ));
13263:   'CS_KEYCVTWINDOW', 'LongInt').SetInt( 4);
13264:   'CS_DBLCLKS', 'LongInt').SetInt( 8);
13265:   'CS_OWNDC', 'LongWord').SetUInt( $20);
13266:   'CS_CLASSDC', 'LongWord').SetUInt( $40);
13267:   'CS_PARENTDC', 'LongWord').SetUInt( $80);
13268:   'CS_NOKEYCVT', 'LongWord').SetUInt( $100);
13269:   'CS_NOCLOSE', 'LongWord').SetUInt( $200);
13270:   'CS_SAVEBITS', 'LongWord').SetUInt( $800);
13271:   'CS_BYTEALIGNCLIENT', 'LongWord').SetUInt( $1000);
13272:   'CS_BYTEALIGNWINDOW', 'LongWord').SetUInt( $2000);
13273:   'CS_GLOBALCLASS', 'LongWord').SetUInt( $4000);
13274:   'CS_IME', 'LongWord').SetUInt( $10000);
13275:   'CS_DROPShadow', 'LongWord').SetUInt( $20000);
13276:   //PPanelFunc', '^TPanelFunc // will not work

```

```

13277: TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13278: TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13279: TFontLooks', 'set of TFontLook
13280: TMessagefunc', 'function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13281: Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13282: Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13283: Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13284: Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13285: Procedure RunMsgLoop( Show : Boolean)
13286: Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13287: Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal;hFont:Int):Int;
13288: Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13289: Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13290: Function MakePanel(Left,Top,Width,Height,
hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13291: Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13292: Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13293: Procedure DoInitMakeApp //set first to init formclasscontrol!
13294: end;
13295:
13296: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13297: begin
13298:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13299:   + 'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13300:   TScreenSaverOptions', 'set of TScreenSaverOption
13301:   'cDefaultScreenSaverOptions', 'LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13302:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13303:   SIRegister_TScreenSaver(CL);
13304:   //Procedure Register
13305:   Procedure SetScreenSaverPassword
13306: end;
13307:
13308: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13309: begin
13310:   FindClass('TOBJECT'),'TXCollection
13311:   SIRegister_EFilerException(CL);
13312:   SIRegister_TXCollectionItem(CL);
13313:   //TXCollectionItemClass', 'class of TXCollectionItem
13314:   SIRegister_TXCollection(CL);
13315:   Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13316:   Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13317:   Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13318:   Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13319:   Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13320:   Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13321: end;
13322:
13323: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13324: begin
13325:   TMapTexCoordMode', '( mtcUndefined, mtcNull, mtcMain, mtcDual, mtcSecond,mtcmArbitrary);
13326:   Procedure xglMapTexCoordToNull
13327:   Procedure xglMapTexCoordToMain
13328:   Procedure xglMapTexCoordToSecond
13329:   Procedure xglMapTexCoordToDual
13330:   Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13331:   Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13332:   Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13333:   Procedure xglBeginUpdate
13334:   Procedure xglEndUpdate
13335:   Procedure xglPushState
13336:   Procedure xglPopState
13337:   Procedure xglForbidSecondTextureUnit
13338:   Procedure xglAllowSecondTextureUnit
13339:   Function xglGetBitWiseMapping : Cardinal
13340: end;
13341:
13342: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13343: begin
13344:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13345:   TBaseListOptions', 'set of TBaseListOption
13346:   SIRegister_TBaseList(CL);
13347:   SIRegister_TBaseVectorList(CL);
13348:   SIRegister_TAffineVectorList(CL);
13349:   SIRegister_TVectorList(CL);
13350:   SIRegister_TTexPointList(CL);
13351:   SIRegister_TXIntegerList(CL);
13352:   //PSingleArrayList', 'TSingleArrayList // will not work
13353:   SIRegister_TSingleList(CL);
13354:   SIRegister_TByteList(CL);
13355:   SIRegister_TQuaternionList(CL);
13356:   Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13357:   Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13358:   Procedure FastQuickSortLists(startIndex,
endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13359: end;
13360:
13361: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);

```

```

13362: begin
13363:   Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13364:   Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13365:   Procedure ConvertStripToList2(const strip:TAffineVectorList;const
indices:TIntegerList;list:TAffineVectorList);
13366:   Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
indices:TIntegerList;list:TAffineVectorList);
13367:   Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13368:   Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13369:   Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13370:   Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13371:   Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13372:   Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13373:   Procedure RemapIndices( indices, indicesMap : TIntegerList)
13374:   Procedure UnifyTrianglesWinding( indices : TIntegerList)
13375:   Procedure InvertTrianglesWinding( indices : TIntegerList)
13376:   Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13377:   Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
edgesTriangles : TIntegerList) : TIntegerList
13378:   Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13379:   Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean);
TPersistentObjectList;
13380:   Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13381:   Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13382: end;
13383:
13384: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13385: begin
13386:   Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13387:   Procedure FreeMemAndNil( var P : TObject)
13388:   Function PCharOrNil( const S : string) : PChar
13389:   SIRegister_TJclReferenceMemoryStream(CL);
13390:   FindClass('TObject','EJclVMError
13391: {Function GetVirtualMethodCount( AClass : TClass) : Integer
13392: Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13393: Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13394:   PDynamicIndexList', '^TDynamicIndexList // will not work
13395:   PDynamicAddressList', '^TDynamicAddressList // will not work
13396: Function GetDynamicMethodCount( AClass : TClass) : Integer
13397: Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13398: Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13399: Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13400: Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13401: Function GetInitTable( AClass : TClass) : PTypeInfo
13402:   PFieldEntry', '^TFieldEntry // will not work}
13403:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : Short'
13404:   + 'String; end
13405:   Function JIsClass( Address : Pointer) : Boolean
13406:   Function JIsObject( Address : Pointer) : Boolean
13407:   Function GetImplementorOfInterface( const I : IInterface) : TObject
13408:   TDigitCount', 'Integer
13409:   SIRegister_TJclNumericFormat(CL);
13410:   Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13411:   TTextHandler', 'Procedure ( const Text : string)
13412: // 'ABORT_EXIT_CODE','LongInt').SetInt( ERROR_CANCELLED 1223);
13413:   Function JExecute(const
CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Card;
13414:   Function JExecutel(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13415:   Function ReadKey : Char //to and from the DOS console !
13416:   TModuleHandle', 'HINST
13417: //TModuleHandle', 'Pointer
13418: 'INVALID_MODULEHANDLE_VALUE','LongInt').SetInt( TModuleHandle ( 0 ));
13419:   Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13420:   Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13421:   Procedure UnloadModule( var Module : TModuleHandle)
13422:   Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13423:   Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13424:   Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13425:   Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13426:   FindClass('TObject','EJclConversionError
13427:   Function JStrToBoolean( const S : string) : Boolean
13428:   Function JBooleanToStr( B : Boolean) : string
13429:   Function JIntToBool( I : Integer) : Boolean
13430:   Function JBoolToInt( B : Boolean) : Integer
13431:   'ListSeparator','String').SetString( ' ;
13432:   'ListSeparator1','String').SetString( ' :
13433:   Procedure ListAddItems( var List : string; const Separator, Items : string)
13434:   Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13435:   Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13436:   Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13437:   Function ListItemCount( const List, Separator : string) : Integer
13438:   Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13439:   Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13440:   Function ListItemIndex( const List, Separator, Item : string) : Integer
13441:   Function SystemToObjectInstance : LongWord
13442:   Function IsCompiledWithPackages : Boolean
13443:   Function JjclGUIDToString( const GUID : TGUID) : string

```



```

13444: Function JJclStringToGUID( const S : string) : TGUID
13445:   SIRegister_TJclIntfCriticalSection(CL);
13446:   SIRegister_TJclSimpleLog(CL);
13447: Procedure InitSimpleLog( const ALogFileName : string)
13448: end;
13449:
13450: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13451: begin
13452:   FindClass('TOBJECT'), 'EJclBorRAException
13453:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13454:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13455:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13456:   TJclBorRADToolPath', string
13457:   'SupportedDelphiVersions', 'LongInt').SetInt( 5 or 6 or 7 or 8 or 9 or 10 or 11);
13458:   'SupportedBCBVersions', 'LongInt').SetInt( 5 or 6 or 10 or 11);
13459:   'SupportedBDSVersions', 'LongInt').SetInt( 1 or 2 or 3 or 4 or 5);
13460:   BorRADToolRepositoryPagesSection', 'String').SetString( 'Repository Pages
13461:   BorRADToolRepositoryDialogsPage', 'String').SetString( 'Dialogs
13462:   BorRADToolRepositoryFormsPage', 'String').SetString( 'Forms
13463:   BorRADToolRepositoryProjectsPage', 'String').SetString( 'Projects
13464:   BorRADToolRepositoryDataModulesPage', 'String').SetString( 'Data Modules
13465:   BorRADToolRepositoryObjectType', 'String').SetString( 'Type
13466:   BorRADToolRepositoryFormTemplate', 'String').SetString( 'FormTemplate
13467:   BorRADToolRepositoryProjectTemplate', 'String').SetString( 'ProjectTemplate
13468:   BorRADToolRepositoryObjectName', 'String').SetString( 'Name
13469:   BorRADToolRepositoryObjectPage', 'String').SetString( 'Page
13470:   BorRADToolRepositoryObjectIcon', 'String').SetString( 'Icon
13471:   BorRADToolRepositoryObjectDescr', 'String').SetString( 'Description
13472:   BorRADToolRepositoryObjectAuthor', 'String').SetString( 'Author
13473:   BorRADToolRepositoryObjectAncestor', 'String').SetString( 'Ancestor
13474:   BorRADToolRepositoryObjectDesigner', 'String').SetString( 'Designer
13475:   BorRADToolRepositoryDesignerDfm', 'String').SetString( 'dfm
13476:   BorRADToolRepositoryDesignerXfm', 'String').SetString( 'xfm
13477:   BorRADToolRepositoryObjectNewForm', 'String').SetString( 'DefaultNewForm
13478:   BorRADToolRepositoryObjectMainForm', 'String').SetString( 'DefaultMainForm
13479:   SourceExtensionDelphiPackage', 'String').SetString( '.dpk
13480:   SourceExtensionBCBPackage', 'String').SetString( '.bpc
13481:   SourceExtensionDelphiProject', 'String').SetString( '.dpr
13482:   SourceExtensionBCBProject', 'String').SetString( '.bpr
13483:   SourceExtensionBDSProject', 'String').SetString( '.bdsproj
13484:   SourceExtensionDProject', 'String').SetString( '.dproj
13485:   BinaryExtensionPackage', 'String').SetString( '.bpl
13486:   BinaryExtensionLibrary', 'String').SetString( '.dll
13487:   BinaryExtensionExecutable', 'String').SetString( '.exe
13488:   CompilerExtensionDCP', 'String').SetString( '.dcp
13489:   CompilerExtensionBPI', 'String').SetString( '.bpi
13490:   CompilerExtensionLIB', 'String').SetString( '.lib
13491:   CompilerExtensionTDS', 'String').SetString( '.tds
13492:   CompilerExtensionMAP', 'String').SetString( '.map
13493:   CompilerExtensionDRC', 'String').SetString( '.drc
13494:   CompilerExtensionDEF', 'String').SetString( '.def
13495:   SourceExtensionCPP', 'String').SetString( '.cpp
13496:   SourceExtensionH', 'String').SetString( '.h
13497:   SourceExtensionPAS', 'String').SetString( '.pas
13498:   SourceExtensionDFM', 'String').SetString( '.dfm
13499:   SourceExtensionXFM', 'String').SetString( '.xfm
13500:   SourceDescriptionPAS', 'String').SetString( 'Pascal source file
13501:   SourceDescriptionCPP', 'String').SetString( 'C++ source file
13502:   DesignerVCL', 'String').SetString( 'VCL
13503:   DesignerCLX', 'String').SetString( 'CLX
13504:   ProjectTypePackage', 'String').SetString( 'package
13505:   ProjectTypeLibrary', 'String').SetString( 'library
13506:   ProjectTypeProgram', 'String').SetString( 'program
13507:   Personality32Bit', 'String').SetString( '32 bit
13508:   Personality64Bit', 'String').SetString( '64 bit
13509:   PersonalityDelphi', 'String').SetString( 'Delphi
13510:   PersonalityDelphiDotNet', 'String').SetString( 'Delphi.net
13511:   PersonalityBCB', 'String').SetString( 'C++Builder
13512:   PersonalityCSB', 'String').SetString( 'C#Builder
13513:   PersonalityVB', 'String').SetString( 'Visual Basic
13514:   PersonalityDesign', 'String').SetString( 'Design
13515:   PersonalityUnknown', 'String').SetString( 'Unknown personality
13516:   PersonalityBDS', 'String').SetString( 'Borland Developer Studio
13517:   DOFDirectoriesSection', 'String').SetString( 'Directories
13518:   DOFUnitOutputDirKey', 'String').SetString( 'UnitOutputDir
13519:   DOFSearchPathName', 'String').SetString( 'SearchPath
13520:   DOFConditionals', 'String').SetString( 'Conditionals
13521:   DOFLinkerSection', 'String').SetString( 'Linker
13522:   DOFPackagesKey', 'String').SetString( 'Packages
13523:   DOFCompilerSection', 'String').SetString( 'Compiler
13524:   DOFPackageNoLinkKey', 'String').SetString( 'PackageNoLink
13525:   DOFAdditionalSection', 'String').SetString( 'Additional
13526:   DOFOptionsKey', 'String').SetString( 'Options
13527:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b
13528:   + 'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13529:   + 'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13530:   TJclBorPersonalities', 'set of TJclBorPersonality
13531:   TJclBorDesigner', '( bdVCL, bdCLX )
13532:   TJclBorDesigners', 'set of TJclBorDesigner

```

```

13533: TJclBorPlatform', '( bp32bit, bp64bit )
13534: FindClass('TOBJECT'), 'TJclBorRADToolInstallation
13535: SIRegister_TJclBorRADToolInstallationObject(CL);
13536: SIRegister_TJclBorlandOpenHelp(CL);
13537: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13538: TJclHelp2Objects', 'set of TJclHelp2Object
13539: SIRegister_TJclHelp2Manager(CL);
13540: SIRegister_TJclBorRADToolIdeTool(CL);
13541: SIRegister_TJclBorRADToolIdePackages(CL);
13542: SIRegister_IJclCommandLineTool(CL);
13543: FindClass('TOBJECT'), 'EJclCommandLineToolError
13544: SIRegister_TJclCommandLineTool(CL);
13545: SIRegister_TJclBorlandCommandLineTool(CL);
13546: SIRegister_TJclBCC32(CL);
13547: SIRegister_TJclDCC32(CL);
13548: TJclDCC', 'TJclDCC32
13549: SIRegister_TJclBpr2Mak(CL);
13550: SIRegister_TJclBorlandMake(CL);
13551: SIRegister_TJclBorRADToolPalette(CL);
13552: SIRegister_TJclBorRADToolRepository(CL);
13553: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13554: TCommandLineTools', 'set of TCommandLineTool
13555: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13556: SIRegister_TJclBorRADToolInstallation(CL);
13557: SIRegister_TJclBCBInstallation(CL);
13558: SIRegister_TJclDelphiInstallation(CL);
13559: SIRegister_TJclDCCIL(CL);
13560: SIRegister_TJclBDSInstallation(CL);
13561: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation ) : Boolean
13562: SIRegister_TJclBorRADToolInstallations(CL);
13563: Function BPLFileName( const BPLPath, PackageFileName : string ) : string
13564: Function BinaryFileName( const OutputPath, ProjectFileName : string ) : string
13565: Function IsDelphiPackage( const FileName : string ) : Boolean
13566: Function IsDelphiProject( const FileName : string ) : Boolean
13567: Function IsBCBPackage( const FileName : string ) : Boolean
13568: Function IsBCBProject( const FileName : string ) : Boolean
13569: Procedure GetDPKFileInfo( const DPRFileName: string; out BinaryExtensio: string; const LibSuffix: PString );
13570: Procedure GetBPRFileInfo( const BPRFileName: string; out BinaryFileName: string; const Descript: PString );
13571: Procedure GetDPKFileInfo( const DPKFileName: string; out RunOnly: Bool; const LibSuffix: PString; const
Descript: PString;
13572: Procedure GetBPKFileInfo( const BPKFileName: string; out RunOnly: Bool; const BinaryFName: PString; const
Descript: PString
13573: function SamePath( const Path1, Path2: string ) : Boolean;
13574: end;
13575:
13576: procedure SIRegister_JclFileUtils_max( CL: TPSPascalCompiler );
13577: begin
13578: 'ERROR_NO_MORE_FILES', 'LongInt' ).SetInt( 18 );
13579: //Function stat64( FileName: PChar; var StatBuffer : TStatBuf64 ) : Integer
13580: //Function fstat64( FileDes: Integer; var StatBuffer : TStatBuf64 ) : Integer
13581: //Function lstat64( FileName: PChar; var StatBuffer : TStatBuf64 ) : Integer
13582: 'LPathSeparator', 'String' ).SetString( '/'
13583: 'LDirDelimiter', 'String' ).SetString( '/'
13584: 'LDirSeparator', 'String' ).SetString( ':'
13585: 'JXPathDevicePrefix', 'String' ).SetString( '\\.\
13586: 'JXPathSeparator', 'String' ).SetString( '\
13587: 'JXDirDelimiter', 'String' ).SetString( '\
13588: 'JXDirSeparator', 'String' ).SetString( ';'
13589: 'JXPathUncPrefix', 'String' ).SetString( '\\
13590: 'faNormalFile', 'LongWord' ).SetUInt( $00000080 );
13591: //faUnixSpecific', ' ' ).SetString( faSymLink );
13592: JXTCompactPath', '( cpCenter, cpEnd )
13593: _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f
13594: + 'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :
13595: + ' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13596: TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13597: WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13598:
13599: Function jxPathAddSeparator( const Path : string ) : string
13600: Function jxPathAddExtension( const Path, Extension : string ) : string
13601: Function jxPathAppend( const Path, Append : string ) : string
13602: Function jxPathBuildRoot( const Drive : Byte ) : string
13603: Function jxPathCanonicalize( const Path : string ) : string
13604: Function jxPathCommonPrefix( const Path1, Path2 : string ) : Integer
13605: Function jxPathCompactPath( const DC:HDC; const Path: string; const Width: Integer; CmpFmt: TCompactPath ): string
13606: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string )
13607: Function jxPathExtractFileDirFixed( const S : string ) : string
13608: Function jxPathExtractFileNameNoExt( const Path : string ) : string
13609: Function jxPathExtractPathDepth( const Path : string; Depth : Integer ) : string
13610: Function jxPathGetDepth( const Path : string ) : Integer
13611: Function jxPathGetLongName( const Path : string ) : string
13612: Function jxPathGetShortName( const Path : string ) : string
13613: Function jxPathGetLongName( const Path : string ) : string
13614: Function jxPathGetShortName( const Path : string ) : string
13615: Function jxPathGetRelativePath( Origin, Destination : string ) : string
13616: Function jxPathGetTempPath : string
13617: Function jxPathIsAbsolute( const Path : string ) : Boolean
13618: Function jxPathIsChild( const Path, Base : string ) : Boolean
13619: Function jxPathIsDiskDevice( const Path : string ) : Boolean

```

```

13620: Function jxPathIsUNC( const Path : string) : Boolean
13621: Function jxPathRemoveSeparator( const Path : string) : string
13622: Function jxPathRemoveExtension( const Path : string) : string
13623: Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13624: Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13625:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13626:   JxTFileListOptions', 'set of TFileListOption
13627:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13628:   TFileHandler', 'Procedure ( const FileName : string)
13629:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13630: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13631: //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
FileMatchFunc:TFileMatchFunc):Bool;
13632: Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13633: Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes:Int):Boolean;
13634: Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13635: Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13636: Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13637: Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
IncludeHiddenDirects:Boolean; const SubDirectoriesMask : string; Abort : TObject; ResolveSymLinks :
Boolean)
13638: Procedure jxCreateEmptyFile( const FileName : string)
13639: Function jxCloseVolume( var Volume : THandle) : Boolean
13640: Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13641: Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13642: Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13643: Function jxDelTree( const Path : string) : Boolean
13644: //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13645: Function jxDiskInDrive( Drive : Char) : Boolean
13646: Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13647: Function jxFileCreateTemp( var Prefix : string) : THandle
13648: Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13649: Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13650: Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13651: Function jxFileExists( const FileName : string) : Boolean
13652: Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13653: Function jxFileRestore( const FileName : string) : Boolean
13654: Function jxGetBackupFileName( const FileName : string) : string
13655: Function jxIsBackupFileName( const FileName : string) : Boolean
13656: Function jxFileGetDisplayName( const FileName : string) : string
13657: Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13658: Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13659: Function jxFileGetSize( const FileName : string) : Int64
13660: Function jxFileGetTempName( const Prefix : string) : string
13661: Function jxFileGetTypeNames( const FileName : string) : string
13662: Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13663: Function jxForceDirectories( Name : string) : Boolean
13664: Function jxGetDirectorySize( const Path : string) : Int64
13665: Function jxGetDriveTypeStr( const Drive : Char) : string
13666: Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13667: Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13668: Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13669: Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13670: Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13671: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
ResolveSymLinks:Boolean):Integer
13672: Function jxGetFileLastWrite( const FName : string) : TFileTime;
13673: Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13674: Function jxGetFileLastAccess( const FName : string) : TFileTime;
13675: Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13676: Function jxGetFileCreation( const FName : string) : TFileTime;
13677: Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13678: Function jxGetFileLastWrite( const FName : string; out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13679: Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13680: Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13681: Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13682: Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13683: Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13684: Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13685: Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13686: Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13687: Function jxGetModulePath( const Module : HMODULE) : string
13688: Function jxGetSizeOfFile( const FileName : string) : Int64;
13689: Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13690: Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13691: Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13692: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13693: Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13694: Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13695: Function jxOpenVolume( const Drive : Char) : THandle
13696: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13697: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13698: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13699: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13700: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13701: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13702: Procedure jxShredFile( const FileName : string; Times : Integer)

```

```

13703: Function jxUnlockVolume( var Handle : THandle) : Boolean
13704: Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13705: Function jxSymbolicLinkTarget( const Name : string) : string
13706: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13707: SIRegister_TJclCustomFileAttrMask(CL);
13708: SIRegister_TJclFileAttributeMask(CL);
13709: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13710: + 'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13711: TFileSearchOptions', 'set of TFileSearchOption
13712: TFileSearchTaskID', 'Integer
13713: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearch'
13714: + 'hTaskID; const Aborted : Boolean)
13715: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13716: SIRegister_IJclFileEnumerator(CL);
13717: SIRegister_TJclFileEnumerator(CL);
13718: Function JxFileSearch : IJclFileEnumerator
13719: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )
13720: JxTFileFlags', 'set of TFileFlag
13721: FindClass('TOBJECT'), 'EJclFileVersionInfoError
13722: SIRegister_TJclFileVersionInfo(CL);
13723: Function jxOSIdentToString( const OSIdent : DWORD) : string
13724: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13725: Function jxVersionResourceAvailable( const FileName : string) : Boolean
13726: TFileVersionFormat', '( vfMajorMinor, vfFull )
13727: Function jxFormatVersionString( const HiV, LoV : Word) : string;
13728: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13729: //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo; VersionFormat: TFileVersionFormat):str;
13730: //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13731: //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
Revision:Word);
13732: //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13733: Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
NotAvailableText : string) : string
13734: SIRegister_TJclTempFileStream(CL);
13735: FindClass('TOBJECT'), 'TJclCustomFileMapping
13736: SIRegister_TJclFileMappingView(CL);
13737: TJclFileMappingRoundOffset', '( rvDown, rvUp )
13738: SIRegister_TJclCustomFileMapping(CL);
13739: SIRegister_TJclFileMapping(CL);
13740: SIRegister_TJclSwapFileMapping(CL);
13741: SIRegister_TJclFileMappingStream(CL);
13742: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13743: //PPCharArray', '^TPCharArray // will not work
13744: SIRegister_TJclMappedTextReader(CL);
13745: SIRegister_TJclFileMaskComparator(CL);
13746: FindClass('TOBJECT'), 'EJclPathError
13747: FindClass('TOBJECT'), 'EJclFileUtilsError
13748: FindClass('TOBJECT'), 'EJclTempFileStreamError
13749: FindClass('TOBJECT'), 'EJclTempFileStreamError
13750: FindClass('TOBJECT'), 'EJclFileMappingError
13751: FindClass('TOBJECT'), 'EJclFileMappingViewError
13752: Function jxPathGetLongName2( const Path : string) : string
13753: Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13754: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13755: Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13756: Function jxWin32RestoreFile( const FileName : string) : Boolean
13757: Function jxSamePath( const Path1, Path2 : string) : Boolean
13758: Procedure jxPathListAddItems( var List : string; const Items : string)
13759: Procedure jxPathListIncludeItems( var List : string; const Items : string)
13760: Procedure jxPathListDelItems( var List : string; const Items : string)
13761: Procedure jxPathListDelItem( var List : string; const Index : Integer)
13762: Function jxPathListItemCount( const List : string) : Integer
13763: Function jxPathListGetItem( const List : string; const Index : Integer) : string
13764: Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13765: Function jxPathListItemIndex( const List, Item : string) : Integer
13766: Function jxParamName(Idx: Int; const Separator: string; const AllowedPrefixChars: string; TrimName: Bool): string;
13767: Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13768: Function jxParamValue1(const SearchName: string; const Separator : string; CaseSensitive : Boolean; const
AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13769: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
AllowedPrefixCharacters : string) : Integer
13770: end;
13771:
13772: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13773: begin
13774: 'UTF8FileHeader', 'String').SetString( #Sef#Sbb#Sbf);
13775: Function lCompareFileNames( const Filename1, Filename2 : string) : integer
13776: Function lCompareFileNamesIgnoreCase( const Filename1, Filename2 : string) : integer
13777: Function lCompareFileNames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13778: Function lCompareFileNames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13779: Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13780: Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13781: Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13782: Procedure lCheckIfFileIsExecutable( const AFilename : string)
13783: Procedure lCheckIfFileIsSymlink( const AFilename : string)
13784: Function lFileIsReadable( const AFilename : string) : boolean
13785: Function lFileIsWritable( const AFilename : string) : boolean
13786: Function lFileIsText( const AFilename : string) : boolean
13787: Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean

```



```

13788: Function lFileIsExecutable( const AFilename : string ) : boolean
13789: Function lFileIsSymlink( const AFilename : string ) : boolean
13790: Function lFileIsHardLink( const AFilename : string ) : boolean
13791: Function lFileSize( const Filename : string ) : int64;
13792: Function lGetFileDescription( const AFilename : string ) : string
13793: Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean ) : string
13794: Function lTryReadAllLinks( const Filename : string ) : string
13795: Function lDirPathExists( const FileName : String ) : Boolean
13796: Function lForceDirectory( DirectoryName : string ) : boolean
13797: Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean ) : boolean
13798: Function lProgramDirectory : string
13799: Function lDirectoryIsWritable( const DirectoryName : string ) : boolean
13800: Function lExtractFileNameOnly( const AFilename : string ) : string
13801: Function lExtractFileNameWithoutExt( const AFilename : string ) : string
13802: Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean ) : integer;
13803: Function lCompareFileExt( const Filename, Ext : string ) : integer;
13804: Function lFilenameIsPascalUnit( const Filename : string ) : boolean
13805: Function lAppendPathDelim( const Path : string ) : string
13806: Function lChompPathDelim( const Path : string ) : string
13807: Function lTrimFilename( const AFilename : string ) : string
13808: Function lCleanAndExpandFilename( const Filename : string ) : string
13809: Function lCleanAndExpandDirectory( const Filename : string ) : string
13810: Function lCreateAbsolutePath( const SearchPath, BaseDirectory : string ) : string
13811: Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
AlwaysRequireSharedBaseFolder : Boolean ) : string
13812: Function lCreateAbsolutePath( const Filename, BaseDirectory : string ) : string
13813: Function lFileIsInPath( const Filename, Path : string ) : boolean
13814: Function lFileIsInDirectory( const Filename, Directory : string ) : boolean
13815: TSearchFileInPathFlag, '( sffDontSearchInBasePath, sffSearchLoUpCase )
13816: TSearchFileInPathFlags, 'set of TSearchFileInPathFlag
13817: 'AllDirectoryEntriesMask', 'String').SetString( '*'
13818: Function lGetAllFilesMask : string
13819: Function lGetExeExt : string
13820: Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
TSearchFileInPathFlags ) : string
13821: Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
TSearchFileInPathFlags ) : TStringList
13822: Function lFindDiskFilename( const Filename : string ) : string
13823: Function lFindDiskFileCaseInsensitive( const Filename : string ) : string
13824: Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13825: Function lGetDarwinSystemFilename( Filename : string ) : string
13826: SIRegister_TFileIterator(CL);
13827: TFileFoundEvent, 'Procedure ( FileIterator : TFileIterator)
13828: TDirectoryFoundEvent, 'Procedure ( FileIterator : TFileIterator)
13829: TDirectoryEnterEvent, 'Procedure ( FileIterator : TFileIterator)
13830: SIRegister_TFileSearcher(CL);
13831: Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13832: Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean ) : TStringList
13833: // TCopyFileFlag, '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13834: // TCopyFileFlags, 'set of TCopyFileFlag
13835: Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags ) : boolean
13836: Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean ) : boolean
13837: Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags ) : Boolean
13838: Function lReadFileToString( const Filename : string ) : string
13839: Function lGetTempFilename( const Directory, Prefix : string ) : string
13840: {Function NeedRTLAnsi : boolean
13841: Procedure SetNeedRTLAnsi( NewValue : boolean)
13842: Function UTF8ToSys( const s : string ) : string
13843: Function SysToUTF8( const s : string ) : string
13844: Function ConsoleToUTF8( const s : string ) : string
13845: Function UTF8ToConsole( const s : string ) : string}
13846: Function FileExistsUTF8( const Filename : string ) : boolean
13847: Function FileAgeUTF8( const FileName : string ) : Longint
13848: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
13849: Function ExpandFileNameUTF8( const FileName : string ) : string
13850: Function ExpandUNCFileNameUTF8( const FileName : string ) : string
13851: Function ExtractShortPathNameUTF8( const FileName : String ) : String
13852: Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13853: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13854: Procedure FindCloseUTF8( var F : TSearchrec)
13855: Function FileSetDateUTF8( const FileName : String; Age : Longint ) : Longint
13856: Function FileGetAttrUTF8( const FileName : String ) : Longint
13857: Function FileSetAttrUTF8( const FileName : String; Attr : longint ) : Longint
13858: Function DeleteFileUTF8( const FileName : String ) : Boolean
13859: Function RenameFileUTF8( const OldName, NewName : String ) : Boolean
13860: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean ) : String
13861: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean
13862: Function GetCurrentDirUTF8 : String
13863: Function SetCurrentDirUTF8( const NewDir : String ) : Boolean
13864: Function CreateDirUTF8( const NewDir : String ) : Boolean
13865: Function RemoveDirUTF8( const Dir : String ) : Boolean
13866: Function ForceDirectoriesUTF8( const Dir : string ) : Boolean
13867: Function FileOpenUTF8( const FileName : string; Mode : Integer ) : THandle
13868: Function FileCreateUTF8( const FileName : string ) : THandle;
13869: Function FileCreateUTF81( const FileName : string; Rights : Cardinal ) : THandle;
13870: Function ParamStrUTF8( Param : Integer ) : string
13871: Function GetEnvironmentStringUTF8( Index : Integer ) : string
13872: Function GetEnvironmentVariableUTF8( const EnvVar : string ) : String
13873: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean ) : string

```

```

13874: Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13875: Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13876: end;
13877:
13878: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13879: begin
13880:     //VK_F23 = 134;
13881:     //{$EXTERNALSYM VK_F24}
13882:     //VK_F24 = 135;
13883:     TVirtualKeyCode, 'Integer
13884:     'VK_MOUSEWHEELUP', 'integer').SetInt(134);
13885:     'VK_MOUSEWHEELDOWN', 'integer').SetInt(135);
13886:     Function glIsKeyDown( c : Char) : Boolean;
13887:     Function glIsKeyDown( vk : TVirtualKeyCode) : Boolean;
13888:     Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13889:     Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13890:     Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13891:     Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13892:     Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13893: end;
13894:
13895: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13896: begin
13897:     TGLPoint, 'TPoint
13898:     //PGLPoint', '^TGLPoint // will not work
13899:     TGLRect, 'TRect
13900:     //PGLRect', '^TGLRect // will not work
13901:     TDelphiColor, 'TColor
13902:     TGLPicture, 'TPicture
13903:     TGLGraphic, 'TGraphic
13904:     TGLBitmap, 'TBitmap
13905:     //TGraphicClass', 'class of TGraphic
13906:     TGLTextLayout, '( tlTop, tlCenter, tlBottom )
13907:     TGLMouseButton, '( mbLeft, mbRight, mbMiddle )
13908:     TGLMouseEvent, 'Procedure ( Sender : TObject; Button : TGLMouse'
13909:         + 'Button; Shift : TShiftState; X, Y : Integer)
13910:     TGLMouseMoveEvent, 'TMouseMoveEvent
13911:     TGLKeyEvent, 'TKeyEvent
13912:     TGLKeyPressEvent, 'TKeyPressEvent
13913:     EGLOSError, 'EWin32Error
13914:     EGLOSError, 'EWin32Error
13915:     EGLOSError, 'EOSError
13916:     'glsAllFilter', 'string').SetString('All // sAllFilter
13917:     Function GLPoint( const x, y : Integer) : TGLPoint
13918:     Function GLRGB( const r, g, b : Byte) : TColor
13919:     Function GLColorToRGB( color : TColor) : TColor
13920:     Function GLGetRValue( rgb : DWORD) : Byte
13921:     Function GLGetGValue( rgb : DWORD) : Byte
13922:     Function GLGetBValue( rgb : DWORD) : Byte
13923:     Procedure GLInitWinColors
13924:     Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13925:     Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13926:     Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13927:     Procedure GLInformationDlg( const msg : String)
13928:     Function GLQuestionDlg( const msg : String) : Boolean
13929:     Function GLInputDialog( const aCaption, aPrompt, aDefault : String) : String
13930:     Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
13931:     Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
13932:     Function GLApplicationTerminated : Boolean
13933:     Procedure GLRaiseLastOSError
13934:     Procedure GLFreeAndNil( var anObject: TObject)
13935:     Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
13936:     Function GLGetCurrentColorDepth : Integer
13937:     Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
13938:     Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
13939:     Procedure GLSleep( length : Cardinal)
13940:     Procedure GLQueryPerformanceCounter( var val : Int64)
13941:     Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
13942:     Function GLStartPrecisionTimer : Int64
13943:     Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
13944:     Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
13945:     Function GLRDTSC : Int64
13946:     Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
13947:     Function GLOKMessageBox( const Text, Caption : string) : Integer
13948:     Procedure GLShowHTMLUrl( Url : String)
13949:     Procedure GLShowCursor( AShow : boolean)
13950:     Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
13951:     Procedure GLGetCursorPos( var point : TGLPoint)
13952:     Function GLGetScreenWidth : integer
13953:     Function GLGetScreenHeight : integer
13954:     Function GLGetTickCount : int64
13955:     function RemoveSpaces(const str : String) : String;
13956:     TNormalMapSpace, '( nmsObject, nmsTangent )
13957:     Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
13958:     Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
13959:     Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
        TAffineVectorList; Colors : TVectorList)
13960:     Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
        HiTexCoords:TAffineVectorList):TGLBitmap

```

```

13961:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
      LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
13962:  end;
13963:
13964:  procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
13965:  begin
13966:    TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
13967:    // PGLStarRecord', '^TGLStarRecord // will not work
13968:    Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
13969:    Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
13970:    Function StarRecordFromColor( const starRecord : TGLStarRecord; bias : Single) : TVector
13971:  end;
13972:
13973:
13974:  procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
13975:  begin
13976:    TAABB', 'record min : TAffineVector; max : TAffineVector; end
13977:    //PAABB', '^TAABB // will not work
13978:    TBSphere', 'record Center : TAffineVector; Radius : single; end
13979:    TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
13980:    TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
13981:    Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
13982:    Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
13983:    Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
13984:    Procedure SetAABB( var bb : TAABB; const v : TVector)
13985:    Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
13986:    Procedure AABBBTransform( var bb : TAABB; const m : TMatrix)
13987:    Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
13988:    Function BBMinX( const c : THmgBoundingBox) : Single
13989:    Function BBMaxX( const c : THmgBoundingBox) : Single
13990:    Function BBMinY( const c : THmgBoundingBox) : Single
13991:    Function BBMaxY( const c : THmgBoundingBox) : Single
13992:    Function BBMinZ( const c : THmgBoundingBox) : Single
13993:    Function BBMaxZ( const c : THmgBoundingBox) : Single
13994:    Procedure AABBBInclude( var bb : TAABB; const p : TAffineVector)
13995:    Procedure AABBFFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
13996:    Function AABBBIntersection( const aabb1, aabb2 : TAABB) : TAABB
13997:    Function BBToAABB( const aabb : THmgBoundingBox) : TAABB
13998:    Function AABBBToBB( const anAABB : TAABB) : THmgBoundingBox;
13999:    Function AABBBToBBI( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14000:    Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14001:    Procedure OffsetAABBI( var aabb : TAABB; const delta : TVector);
14002:    Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14003:    Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14004:    Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14005:    Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14006:    Function AABBBFitsInAABBBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14007:    Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14008:    Function PointInAABBI( const p : TVector; const aabb : TAABB) : Boolean;
14009:    Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14010:    Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14011:    Procedure ExtractAABBCorners( const AABBB : TAABB; var AABBCorners : TAABBCorners)
14012:    Procedure AABBBToBSphere( const AABBB : TAABB; var BSphere : TBSphere)
14013:    Procedure BSphereToAABB( const BSphere : TBSphere; var AABBB : TAABB);
14014:    Function BSphereToAABBI( const center : TAffineVector; radius : Single) : TAABB;
14015:    Function BSphereToAABBI2( const center : TVector; radius : Single) : TAABB;
14016:    Function AABBBContainsAABBB( const mainAABBB, testAABBB : TAABB) : TSpaceContains
14017:    Function BSphereContainsAABBB( const mainBSphere : TBSphere; const testAABBB : TAABB) : TSpaceContains
14018:    Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14019:    Function AABBBContainsBSphere( const mainAABBB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14020:    Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
      testBSphere:TBSphere):TSpaceContains
14021:    Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14022:    Function FrustumContainsAABBB( const Frustum : TFrustum; const testAABBB : TAABB) : TSpaceContains
14023:    Function ClipToAABBB( const v : TAffineVector; const AABBB : TAABB) : TAffineVector
14024:    Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14025:    Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14026:    Function AABBBToClipRect(const aabb:TAABBB;modelViewProjection:TMatrix;viewportSizeX,
      viewportSizeY:rint):TClipRect
14027:  end;
14028:
14029:  procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14030:  begin
14031:    Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14032:    Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14033:    Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14034:    Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14035:    Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14036:    Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14037:    Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14038:    Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14039:    Procedure Spherical_Cartesian2( const r,theta,phi : single; var x, y, z : single; var ierr : integer);
14040:    Procedure Spherical_Cartesian3( const r,theta,phi : double; var x, y, z : double; var ierr : integer);
14041:    Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14042:    Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14043:    Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14044:    Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14045:    Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14046:    Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);

```

```

14047: Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14048: Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14049: Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14050: Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14051: Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14052: Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14053: Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14054: Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14055: Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14056: end;
14057:
14058: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14059: begin
14060:   'EPSILON','Single').setExtended( 1e-40);
14061:   'EPSILON2','Single').setExtended( 1e-30); }
14062: TRenderContextClippingInfo, 'record origin : TVector; clippingD'
14063:   + 'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14064: THmgPlane', 'TVector
14065:   TDoubleHmgPlane', 'THomogeneousDblVector
14066: {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14067:   + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14068:   + ', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW }}
14069: TSingleArray', 'array of Single
14070: TTransformations', 'array [0..15] of Single)
14071: TPackedRotationMatrix', 'array [0..2] of Smallint)
14072: TVertex', 'TAffineVector
14073: //TVectorGL', 'THomogeneousFltVector
14074: //TMatrixGL', 'THomogeneousFltMatrix
14075: // TPackedRotationMatrix = array [0..2] of SmallInt;
14076: Function glTexPointMake( const s, t : Single) : TTexPoint
14077: Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14078: Function glAffineVectorMakel( const v : TVectorGL) : TAffineVector;
14079: Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14080: Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14081: Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14082: Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14083: Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14084: Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14085: Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14086: Function glVectorMakel( const x, y, z : Single; w : Single) : TVectorGL;
14087: Function glPointMake( const x, y, z : Single) : TVectorGL;
14088: Function glPointMakel( const v : TAffineVector) : TVectorGL;
14089: Function glPointMake2( const v : TVectorGL) : TVectorGL;
14090: Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14091: Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14092: Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14093: Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14094: Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14095: Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14096: Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14097: Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14098: Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14099: Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14100: Procedure glRstVector( var v : TAffineVector);
14101: Procedure glRstVector1( var v : TVectorGL);
14102: Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14103: Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14104: //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14105: Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14106: Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14107: Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14108: Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14109: Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14110: Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14111: Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14112: Procedure glAddVector10( var v : TAffineVector; const f : Single);
14113: Procedure glAddVector11( var v : TVectorGL; const f : Single);
14114: //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Int;dest:PTexPointArray);
14115: //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14116: //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
PAffineVectorArray);
14117: Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14118: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14119: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14120: Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14121: Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14122: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14123: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14124: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14125: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14126: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14127: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14128: Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14129: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14130: Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14131: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14132: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;

```



```

14133: Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14134: Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14135: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14136: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14137: Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14138: Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14139: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14140: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14141: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14142: Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14143: Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14144: Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14145: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14146: Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14147: Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14148: Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14149: Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14150: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14151: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14152: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14153: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14154: Function glLerp( const start, stop, t : Single) : Single
14155: Function glAngleLerp( start, stop, t : Single) : Single
14156: Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14157: Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14158: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14159: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14160: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14161: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14162: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14163: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14164: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14165: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
PAffineVectorArray);
14166: Function glVectorLength( const x, y : Single) : Single;
14167: Function glVectorLength1( const x, y, z : Single) : Single;
14168: Function glVectorLength2( const v : TAffineVector) : Single;
14169: Function glVectorLength3( const v : TVectorGL) : Single;
14170: Function glVectorLength4( const v : array of Single) : Single;
14171: Function glVectorNorm( const x, y : Single) : Single;
14172: Function glVectorNorm1( const v : TAffineVector) : Single;
14173: Function glVectorNorm2( const v : TVectorGL) : Single;
14174: Function glVectorNorm3( var V : array of Single) : Single;
14175: Procedure glNormalizeVector( var v : TAffineVector);
14176: Procedure glNormalizeVector1( var v : TVectorGL);
14177: Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14178: Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14179: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14180: Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14181: Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14182: Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14183: Procedure glNegateVector( var V : TAffineVector);
14184: Procedure glNegateVector2( var V : TVectorGL);
14185: Procedure glNegateVector3( var V : array of Single);
14186: Procedure glScaleVector( var v : TAffineVector; factor : Single);
14187: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14188: Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14189: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14190: Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14191: Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14192: Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14193: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14194: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14195: Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14196: Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14197: Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14198: Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14199: Function glVectorIsNull( const v : TVectorGL) : Boolean;
14200: Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14201: Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14202: Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14203: Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14204: Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14205: Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14206: Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14207: Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14208: Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14209: Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14210: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14211: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14212: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14213: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14214: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14215: Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14216: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14217: Procedure glAbsVector( var v : TVectorGL);
14218: Procedure glAbsVector1( var v : TAffineVector);
14219: Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14220: Function glVectorAbs1( const v : TAffineVector) : TAffineVector;

```

```

14221: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14222: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14223: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14224: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14225: Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14226: Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14227: Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14228: Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14229: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14230: Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14231: Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14232: Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14233: Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14234: Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14235: Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14236: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14237: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14238: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14239: Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14240: Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14241: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14242: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14243: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14244: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14245: Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14246: Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14247: Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14248: Procedure glAdjointMatrix( var M : TMatrixGL);
14249: Procedure glAdjointMatrix1( var M : TAffineMatrix);
14250: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14251: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14252: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14253: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14254: Procedure glNormalizeMatrix( var M : TMatrixGL);
14255: Procedure glTransposeMatrix( var M : TAffineMatrix);
14256: Procedure glTransposeMatrix1( var M : TMatrixGL);
14257: Procedure glInvertMatrix( var M : TMatrixGL);
14258: Procedure glInvertMatrix1( var M : TAffineMatrix);
14259: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14260: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14261: Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14262: Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14263: Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14264: Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14265: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14266: Procedure glNormalizePlane( var plane : THmgPlane)
14267: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14268: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14269: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14270: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14271: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14272: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14273: Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14274: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14275: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14276: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14277: Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14278: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14279: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14280: Procedure SgsegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
Segment0Closest, Segment1Closest : TAffineVector)
14281: Function glSegmentSegmentDistance( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector) : single
14282: TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14283: Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14284: Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14285: Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14286: Procedure glNormalizeQuaternion( var Q : TQuaternion)
14287: Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14288: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14289: Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14290: Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14291: Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14292: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14293: Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14294: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14295: Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14296: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14297: Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14298: Function glLnXP1( X : Extended) : Extended
14299: Function glLog10( X : Extended) : Extended
14300: Function glLog2( X : Extended) : Extended;
14301: Function glLog21( X : Single) : Single;
14302: Function glLogN( Base, X : Extended) : Extended
14303: Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14304: Function glPower( const Base, Exponent : Single) : Single;
14305: Function glPower1( Base : Single; Exponent : Integer) : Single;
14306: Function glDegToRad( const Degrees : Extended) : Extended;
14307: Function glDegToRad1( const Degrees : Single) : Single;
14308: Function glRadToDeg( const Radians : Extended) : Extended;

```

```

14309: Function glRadToDeg1( const Radians : Single) : Single;
14310: Function glNormalizeAngle( angle : Single) : Single
14311: Function glNormalizeDegAngle( angle : Single) : Single
14312: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14313: Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14314: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14315: Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14316: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14317: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14318: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14319: Function glArcCos( const X : Extended) : Extended;
14320: Function glArcCos1( const x : Single) : Single;
14321: Function glArcSin( const X : Extended) : Extended;
14322: Function glArcSin1( const X : Single) : Single;
14323: Function glArcTan21( const Y, X : Extended) : Extended;
14324: Function glArcTan21( const Y, X : Single) : Single;
14325: Function glFastArcTan2( y, x : Single) : Single
14326: Function glTan( const X : Extended) : Extended;
14327: Function glTan1( const X : Single) : Single;
14328: Function glCoTan( const X : Extended) : Extended;
14329: Function glCoTan1( const X : Single) : Single;
14330: Function glSinh( const x : Single) : Single;
14331: Function glSinh1( const x : Double) : Double;
14332: Function glCosh( const x : Single) : Single;
14333: Function glCosh1( const x : Double) : Double;
14334: Function glRSqrt( v : Single) : Single
14335: Function glRLength( x, y : Single) : Single
14336: Function glISqrt( i : Integer) : Integer
14337: Function glILength( x, y : Integer) : Integer;
14338: Function glILength1( x, y, z : Integer) : Integer;
14339: Procedure glRegisterBasedExp
14340: Procedure glRandomPointOnSphere( var p : TAffineVector)
14341: Function glRoundInt( v : Single) : Single;
14342: Function glRoundInt1( v : Extended) : Extended;
14343: Function glTrunc( v : Single) : Integer;
14344: Function glTrunc64( v : Extended) : Int64;
14345: Function glInt( v : Single) : Single;
14346: Function glInt1( v : Extended) : Extended;
14347: Function glFrac( v : Single) : Single;
14348: Function glFrac1( v : Extended) : Extended;
14349: Function glRound( v : Single) : Integer;
14350: Function glRound64( v : Single) : Int64;
14351: Function glRound641( v : Extended) : Int64;
14352: Function glTrunc( X : Extended) : Int64
14353: Function glRound( X : Extended) : Int64
14354: Function glFrac( X : Extended) : Extended
14355: Function glCeil( v : Single) : Integer;
14356: Function glCeil64( v : Extended) : Int64;
14357: Function glFloor( v : Single) : Integer;
14358: Function glFloor64( v : Extended) : Int64;
14359: Function glScaleAndRound( i : Integer; var s : Single) : Integer
14360: Function glSign( x : Single) : Integer
14361: Function glIsInRange( const x, a, b : Single) : Boolean;
14362: Function glIsInRangel( const x, a, b : Double) : Boolean;
14363: Function glIsInCube( const p, d : TAffineVector) : Boolean;
14364: Function glIsInCubel( const p, d : TVectorGL) : Boolean;
14365: //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14366: //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14367: //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14368: Function glMinFloat3( const v1, v2 : Single) : Single;
14369: Function glMinFloat4( const v : array of Single) : Single;
14370: Function glMinFloat5( const v1, v2 : Double) : Double;
14371: Function glMinFloat6( const v1, v2 : Extended) : Extended;
14372: Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14373: Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14374: Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14375: //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14376: //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14377: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14378: Function glMaxFloat2( const v : array of Single) : Single;
14379: Function glMaxFloat3( const v1, v2 : Single) : Single;
14380: Function glMaxFloat4( const v1, v2 : Double) : Double;
14381: Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14382: Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14383: Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14384: Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14385: Function glMinInteger9( const v1, v2 : Integer) : Integer;
14386: Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14387: Function glMaxInteger( const v1, v2 : Integer) : Integer;
14388: Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14389: Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14390: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14391: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14392: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14393: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14394: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14395: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14396: Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14397: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);

```

```

14398: Function glMaxXYZComponent( const v : TVectorGL ) : Single;
14399: Function glMaxXYZComponent1( const v : TAffineVector ) : single;
14400: Function glMinXYZComponent( const v : TVectorGL ) : Single;
14401: Function glMinXYZComponent1( const v : TAffineVector ) : single;
14402: Function glMaxAbsXYZComponent( v : TVectorGL ) : Single
14403: Function glMinAbsXYZComponent( v : TVectorGL ) : Single
14404: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14405: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14406: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14407: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14408: Procedure glSortArrayAscending( var a : array of Extended)
14409: Function glClampValue( const aValue, aMin, aMax : Single ) : Single;
14410: Function glClampValue1( const aValue, aMin : Single ) : Single;
14411: Function glGeometryOptimizationMode : String
14412: Procedure glBeginFPUOnlySection
14413: Procedure glEndFPUOnlySection
14414: Function glConvertRotation( const Angles : TAffineVector ) : TVectorGL
14415: Function glMakeAffineDblVector( var v : array of Double ) : TAffineDblVector
14416: Function glMakeDblVector( var v : array of Double ) : THomogeneousDblVector
14417: Function glVectorAffineDblToFlt( const v : TAffineDblVector ) : TAffineVector
14418: Function glVectorDblToFlt( const v : THomogeneousDblVector ) : THomogeneousVector
14419: Function glVectorAffineFltToDbl( const v : TAffineVector ) : TAffineDblVector
14420: Function glVectorFltToDbl( const v : TVectorGL ) : THomogeneousDblVector
14421: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single ) : Boolean
14422: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14423: Function glTurn( const Matrix : TMatrixGL; angle : Single ) : TMatrixGL;
14424: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14425: Function glPitch( const Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
14426: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14427: Function glRoll( const Matrix: TMatrixGL; Angle : Single ) : TMatrixGL;
14428: Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14429: Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14430: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
sphereCenter:TVectorGL;const sphereRadius : Single ) : Boolean;
14431: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
const sphereRadius : Single; var i1, i2 : TVectorGL ) : Integer;
14432: Function glSphereVisibleRadius( distance, radius : Single ) : Single
14433: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL ) : TFrustum
14434: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
TRenderContextClippingInfo ) : Boolean;
14435: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
TRenderContextClippingInfo ) : Boolean;
14436: Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo ) : Bool;
14437: Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
Frustum:TFrustum):Bool;
14438: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL ) : TMatrixGL
14439: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL ) : TMatrixGL
14440: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector ) : TMatrixGL
14441: Function glPackRotationMatrix( const mat : TMatrixGL ) : TPackedRotationMatrix
14442: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix ) : TMatrixGL
14443: 'cPI','Single').setExtended( 3.141592654);
14444: 'cPIdiv180','Single').setExtended( 0.017453292);
14445: 'c180divPI','Single').setExtended( 57.29577951);
14446: 'c2PI','Single').setExtended( 6.283185307);
14447: 'cPIdiv2','Single').setExtended( 1.570796326);
14448: 'cPIdiv4','Single').setExtended( 0.785398163);
14449: 'c3PIdiv4','Single').setExtended( 2.35619449);
14450: 'cInv2PI','Single').setExtended( 1 / 6.283185307);
14451: 'cInv360','Single').setExtended( 1 / 360);
14452: 'c180','Single').setExtended( 180);
14453: 'c360','Single').setExtended( 360);
14454: 'cOneHalf','Single').setExtended( 0.5);
14455: 'cLn10','Single').setExtended( 2.302585093);
14456: {'MinSingle','Extended').setExtended( 1.5e-45);
14457: 'MaxSingle','Extended').setExtended( 3.4e+38);
14458: 'MinDouble','Extended').setExtended( 5.0e-324);
14459: 'MaxDouble','Extended').setExtended( 1.7e+308);
14460: 'MinExtended','Extended').setExtended( 3.4e-4932);
14461: 'MaxExtended','Extended').setExtended( 1.1e+4932);
14462: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14463: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14464: end;
14465:
14466: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14467: begin
14468:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14469:   (FindClass('TOBJECT'),'TFaceGroups
14470:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14471:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14472:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14473:   SIRegister_TBaseMeshObject(CL);
14474:   (FindClass('TOBJECT'),'TSkeletonFrameList
14475:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14476:   SIRegister_TSkeletonFrame(CL);
14477:   SIRegister_TSkeletonFrameList(CL);
14478:   (FindClass('TOBJECT'),'TSkeleton
14479:   (FindClass('TOBJECT'),'TSkeletonBone
14480:   SIRegister_TSkeletonBoneList(CL);
14481:   SIRegister_TSkeletonRootBoneList(CL);

```



```

14482: SIRegister_TSkeletonBone(CL);
14483: (FindClass('TOBJECT'),'TSkeletonColliderList
14484: SIRegister_TSkeletonCollider(CL);
14485: SIRegister_TSkeletonColliderList(CL);
14486: (FindClass('TOBJECT'),'TGLBaseMesh
14487: TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14488:   +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14489:   +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14490:   +'QuaternionList; end
14491: SIRegister_TSkeleton(CL);
14492: TMeshObjectRenderingOption', '( moroGroupByMaterial )
14493: TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14494: SIRegister_TMeshObject(CL);
14495: SIRegister_TMeshObjectList(CL);
14496: //TMeshObjectListClass', 'class of TMeshObjectList
14497: (FindClass('TOBJECT'),'TMeshMorphTargetList
14498: SIRegister_TMeshMorphTarget(CL);
14499: SIRegister_TMeshMorphTargetList(CL);
14500: SIRegister_TMorphableMeshObject(CL);
14501: TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14502: //TVertexBoneWeightArray', '^TVertexBoneWeightArray // will not work
14503: //TVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14504: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14505: SIRegister_TSkeletonMeshObject(CL);
14506: SIRegister_TFaceGroup(CL);
14507: TFaceGroupMeshMode', '( fgmTriangles, fgmTriangleStrip, fgmFl'
14508:   +'atTriangles, fgmTriangleFan, fgmQuads )
14509: SIRegister_TFGVertexIndexList(CL);
14510: SIRegister_TFGVertexNormalTexIndexList(CL);
14511: SIRegister_TFGIndexTexCoordList(CL);
14512: SIRegister_TFaceGroups(CL);
14513: TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14514: SIRegister_TVectorFile(CL);
14515: //TVectorFileClass', 'class of TVectorFile
14516: SIRegister_TGLGLSMVectorFile(CL);
14517: SIRegister_TGLBaseMesh(CL);
14518: SIRegister_TGLFreeForm(CL);
14519: TGLActorOption', '( aoSkeletonNormalizeNormals )
14520: TGLActorOptions', 'set of TGLActorOption
14521: 'cDefaultGLActorOptions', 'LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14522: (FindClass('TOBJECT'),'TGLActor
14523: TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14524: SIRegister_TActorAnimation(CL);
14525: TActorAnimationName', 'String
14526: SIRegister_TActorAnimations(CL);
14527: SIRegister_TGLBaseAnimationController(CL);
14528: SIRegister_TGLAnimationController(CL);
14529: TActorFrameInterpolation', '( afpNone, afpLinear )
14530: TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14531:   +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14532: SIRegister_TGLActor(CL);
14533: SIRegister_TVectorFileFormat(CL);
14534: SIRegister_TVectorFileFormatsList(CL);
14535: (FindClass('TOBJECT'),'EInvalidVectorFile
14536: Function GetVectorFileFormats : TVectorFileFormatsList
14537: Function VectorFileFormatsFilter : String
14538: Function VectorFileFormatsSaveFilter : String
14539: Function VectorFileFormatExtensionByIndex( index : Integer) : String
14540: Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14541: Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14542: end;
14543:
14544: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14545: begin
14546: 'Class_DColorPropPage', 'TGUID').SetString( '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}'
14547: 'Class_DFontPropPage', 'TGUID').SetString( '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}'
14548: 'Class_DPicturePropPage', 'TGUID').SetString( '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}'
14549: 'Class_DStringPropPage', 'TGUID').SetString( '{F42D677E-754B-11D0-BDFB-00A024D1875C}'
14550: SIRegister_ToleStream(CL);
14551: (FindClass('TOBJECT'),'TConnectionPoints
14552: TConnectionKind', '( ckSingle, ckMulti )
14553: SIRegister_TConnectionPoint(CL);
14554: SIRegister_TConnectionPoints(CL);
14555: TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14556: (FindClass('TOBJECT'),'TActiveXControlFactory
14557: SIRegister_TActiveXControl(CL);
14558: //TActiveXControlClass', 'class of TActiveXControl
14559: SIRegister_TActiveXControlFactory(CL);
14560: SIRegister_TActiveFormControl(CL);
14561: SIRegister_TActiveForm(CL);
14562: //TActiveFormClass', 'class of TActiveForm
14563: SIRegister_TActiveFormFactory(CL);
14564: (FindClass('TOBJECT'),'TPropertyPageImpl
14565: SIRegister_TPropertyPage(CL);
14566: //TPropertyPageClass', 'class of TPropertyPage
14567: SIRegister_TPropertyPageImpl(CL);
14568: SIRegister_TActiveXPropertyPage(CL);
14569: SIRegister_TActiveXPropertyPageFactory(CL);
14570: SIRegister_TCustomAdapter(CL);

```

```

14571:   SIRegister_TAdapterNotifier(CL);
14572:   SIRegister_IFontAccess(CL);
14573:   SIRegister_TFontAdapter(CL);
14574:   SIRegister_IPictureAccess(CL);
14575:   SIRegister_TPictureAdapter(CL);
14576:   SIRegister_TOLEGraphic(CL);
14577:   SIRegister_TStringsAdapter(CL);
14578:   SIRegister_TReflectorWindow(CL);
14579: Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VtCode:Int;PropList:TStrings);
14580: Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14581: Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14582: Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14583: Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14584: Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14585: Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14586: Function ParkingWindow : HWND
14587: end;
14588:
14589: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14590: begin
14591:   // Tip6Bytes = array [0..15] of Byte;
14592:   {:binary form of IPv6 address (for string conversion routines)}
14593:   // Tip6Words = array [0..7] of Word;
14594:   AddTypeS('Tip6Bytes', 'array [0..15] of Byte;');
14595:   AddTypeS('Tip6Words', 'array [0..7] of Word;');
14596:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4 : Byte; end;');
14597:   AddDelphiFunction('Function synaIsIP( const Value : string) : Boolean;');
14598:   Function synaIsIP6( const Value : string) : Boolean;
14599:   Function synaIPToID( Host : string) : AnsiString;
14600:   Function synaStrToIP6( value : string) : Tip6Bytes;
14601:   Function synaIP6ToStr( value : Tip6Bytes) : string;
14602:   Function synaStrToIP( value : string) : integer;
14603:   Function synaIPToStr( value : integer) : string;
14604:   Function synaReverseIP( Value : AnsiString) : AnsiString;
14605:   Function synaReverseIP6( Value : AnsiString) : AnsiString;
14606:   Function synaExpandIP6( Value : AnsiString) : AnsiString;
14607:   Function xStrToIP( const Value : String) : TIPAdr;
14608:   Function xIPToStr( const Adresse : TIPAdr) : String;
14609:   Function IPToCardinal( const Adresse : TIPAdr) : Cardinal;
14610:   Function CardinalToIP( const Value : Cardinal) : TIPAdr;
14611: end;
14612:
14613: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14614: begin
14615:   AddTypeS('TSpecials', 'set of Char');
14616:   Const ('SpecialChar', 'TSpecials').SetSet( '=[<>:;,@/?\"_');
14617:   Const ('URLFullSpecialChar', 'TSpecials').SetSet( ';/?:@=&#+');
14618:   Const ('TableBase64', 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14619:   Const ('TableBase64mod', 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,='');
14620:   Const ('TableUU', '(!#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNQRSTUUVWXYZ[^\_');
14621:   Const ('TableXX', '(+0123456789ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvwxyz');
14622:   Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString;
14623:   Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString;
14624:   Function DecodeURL( const Value : AnsiString) : AnsiString;
14625:   Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString;
14626:   Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString;
14627:   Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString;
14628:   Function EncodeURLElement( const Value : AnsiString) : AnsiString;
14629:   Function EncodeURL( const Value : AnsiString) : AnsiString;
14630:   Function Decode4to3( const Value, Table : AnsiString) : AnsiString;
14631:   Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString;
14632:   Function Encode3to4( const Value, Table : AnsiString) : AnsiString;
14633:   Function synDecodeBase64( const Value : AnsiString) : AnsiString;
14634:   Function synEncodeBase64( const Value : AnsiString) : AnsiString;
14635:   Function DecodeBase64mod( const Value : AnsiString) : AnsiString;
14636:   Function EncodeBase64mod( const Value : AnsiString) : AnsiString;
14637:   Function DecodeUU( const Value : AnsiString) : AnsiString;
14638:   Function EncodeUU( const Value : AnsiString) : AnsiString;
14639:   Function DecodeXX( const Value : AnsiString) : AnsiString;
14640:   Function DecodeYEnc( const Value : AnsiString) : AnsiString;
14641:   Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer;
14642:   Function synCrc32( const Value : AnsiString) : Integer;
14643:   Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word;
14644:   Function Crc16( const Value : AnsiString) : Word;
14645:   Function synMD5( const Value : AnsiString) : AnsiString;
14646:   Function HMAC_MD5( Text, Key : AnsiString) : AnsiString;
14647:   Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString;
14648:   Function synSHA1( const Value : AnsiString) : AnsiString;
14649:   Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString;
14650:   Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString;
14651:   Function synMD4( const Value : AnsiString) : AnsiString;
14652: end;
14653:
14654: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14655: begin
14656:   AddTypeS('TTimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14657:     + 'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14658:     + 'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14659:     + '4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'

```

```

14660:   +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14661:   +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14662:   +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14663:   +', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14664:   +'IS620, CP874, VISCI, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14665:   +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14666:   +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14667:   +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14668:   +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14669:   +', CP864, CP865, CP869, CP1125 )'';
14670:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14671:   Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14672:   Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
TransformTable : array of Word) : AnsiString);
14673:   Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
TransformTable : array of Word; Translit : Boolean) : AnsiString);
14674:   Function GetCurCP : TMimeChar);
14675:   Function GetCurOEMCP : TMimeChar);
14676:   Function GetCPFromID( Value : AnsiString) : TMimeChar);
14677:   Function GetIDFromCP( Value : TMimeChar) : AnsiString);
14678:   Function NeedCharsetConversion( const Value : AnsiString) : Boolean);
14679:   Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14680:   Function GetBOM( Value : TMimeChar) : AnsiString);
14681:   Function StringToWide( const Value : AnsiString) : WideString);
14682:   Function WideToString( const Value : WideString) : AnsiString);
14683: end;
14684:
14685: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14686: begin
14687:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14688:   Procedure WakeOnLan( MAC, IP : string);
14689:   Function GetDNS : string);
14690:   Function GetIEProxy( protocol : string) : TProxySetting);
14691:   Function GetLocalIPs : string);
14692: end;
14693:
14694:
14695: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14696: begin
14697:   AddConstantN('synCR', 'Char').SetString( #$0d);
14698:   Const('synLF', 'Char').SetString( #$0a);
14699:   Const('cSerialChunk', 'LongInt').SetInt( 8192);
14700:   Const('LockfileDirectory', 'String').SetString( '/var/lock');
14701:   Const('PortIsClosed', 'LongInt').SetInt( - 1);
14702:   Const('ErrAlreadyOwned', 'LongInt').SetInt( 9991);
14703:   Const('ErrAlreadyInUse', 'LongInt').SetInt( 9992);
14704:   Const('ErrWrongParameter', 'LongInt').SetInt( 9993);
14705:   Const('ErrPortNotOpen', 'LongInt').SetInt( 9994);
14706:   Const('ErrNoDeviceAnswer', 'LongInt').SetInt( 9995);
14707:   Const('ErrMaxBuffer', 'LongInt').SetInt( 9996);
14708:   Const('ErrTimeout', 'LongInt').SetInt( 9997);
14709:   Const('ErrNotRead', 'LongInt').SetInt( 9998);
14710:   Const('ErrFrame', 'LongInt').SetInt( 9999);
14711:   Const('ErrOverrun', 'LongInt').SetInt( 10000);
14712:   Const('ErrRxOver', 'LongInt').SetInt( 10001);
14713:   Const('ErrRxParity', 'LongInt').SetInt( 10002);
14714:   Const('ErrTxFull', 'LongInt').SetInt( 10003);
14715:   Const('dcb_Binary', 'LongWord').SetUInt( $00000001);
14716:   Const('dcb_ParityCheck', 'LongWord').SetUInt( $00000002);
14717:   Const('dcb_OutxCtsFlow', 'LongWord').SetUInt( $00000004);
14718:   Const('dcb_OutxDsrFlow', 'LongWord').SetUInt( $00000008);
14719:   Const('dcb_DtrControlMask', 'LongWord').SetUInt( $00000030);
14720:   Const('dcb_DtrControlDisable', 'LongWord').SetUInt( $00000000);
14721:   Const('dcb_DtrControlEnable', 'LongWord').SetUInt( $00000010);
14722:   Const('dcb_DtrControlHandshake', 'LongWord').SetUInt( $00000020);
14723:   Const('dcb_DsrSensitivity', 'LongWord').SetUInt( $00000040);
14724:   Const('dcb_TXContinueOnXoff', 'LongWord').SetUInt( $00000080);
14725:   Const('dcb_OutX', 'LongWord').SetUInt( $00000100);
14726:   Const('dcb_InX', 'LongWord').SetUInt( $00000200);
14727:   Const('dcb_ErrorChar', 'LongWord').SetUInt( $00000400);
14728:   Const('dcb_NullStrip', 'LongWord').SetUInt( $00000800);
14729:   Const('dcb_RtsControlMask', 'LongWord').SetUInt( $00003000);
14730:   Const('dcb_RtsControlDisable', 'LongWord').SetUInt( $00000000);
14731:   Const('dcb_RtsControlEnable', 'LongWord').SetUInt( $00001000);
14732:   Const('dcb_RtsControlHandshake', 'LongWord').SetUInt( $00002000);
14733:   Const('dcb_RtsControlToggle', 'LongWord').SetUInt( $00003000);
14734:   Const('dcb_AbortOnError', 'LongWord').SetUInt( $00004000);
14735:   Const('dcb_Reserveds', 'LongWord').SetUInt( $FFFF8000);
14736:   Const('synSB1', 'LongInt').SetInt( 0);
14737:   Const('SBandHalf', 'LongInt').SetInt( 1);
14738:   Const('synSB2', 'LongInt').SetInt( 2);
14739:   Const('synINVALID_HANDLE_VALUE', 'LongInt').SetInt( THandle ( - 1 ));
14740:   Const('CS7fix', 'LongWord').SetUInt( $0000020);
14741:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14742:   + 'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14743:   + 'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14744:   + 'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14745:   //AddTypeS('PDCB', '^TDCB // will not work');
14746:   //Const('MaxRates', 'LongInt').SetInt( 18);

```

```

14747: //Const('MaxRates','LongInt').SetInt( 30);
14748: //Const('MaxRates','LongInt').SetInt( 19);
14749: Const('O_SYNC','LongWord').SetUInt( $0080);
14750: Const('synOK','LongInt').SetInt( 0);
14751: Const('synErr','LongInt').SetInt( integer ( - 1 ));
14752: AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
HR_WriteCount, HR_Wait )');
14753: Type('THookSerialStatus', Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string));
14754: SIRegister_ESynaSerError(CL);
14755: SIRegister_TBlockSerial(CL);
14756: Function GetSerialPortNames : string';
14757: end;
14758:
14759: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14760: begin
14761: Const('DLLIconvName','String').SetString( 'libiconv.so');
14762: Const('DLLIconvName','String').SetString( 'iconv.dll');
14763: AddTypeS('size_t', 'Cardinal');
14764: AddTypeS('iconv_t', 'Integer');
14765: //AddTypeS('iconv_t', 'Pointer');
14766: AddTypeS('argptr', 'iconv_t');
14767: Function SynaIconvOpen( const tocode, fromcode : AnsiString) : iconv_t';
14768: Function SynaIconvOpenTranslit( const tocode, fromcode : AnsiString) : iconv_t';
14769: Function SynaIconvOpenIgnore( const tocode, fromcode : AnsiString) : iconv_t';
14770: Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer';
14771: Function SynaIconvClose( var cd : iconv_t) : integer';
14772: Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer';
14773: Function IsIconvloaded : Boolean';
14774: Function InitIconvInterface : Boolean';
14775: Function DestroyIconvInterface : Boolean';
14776: Const('ICONV_TRIVIALP','LongInt').SetInt( 0);
14777: Const('ICONV_GET_TRANSLITERATE','LongInt').SetInt( 1);
14778: Const('ICONV_SET_TRANSLITERATE','LongInt').SetInt( 2);
14779: Const('ICONV_GET_DISCARD_ILSEQ','LongInt').SetInt( 3);
14780: Const('ICONV_SET_DISCARD_ILSEQ','LongInt').SetInt( 4);
14781: end;
14782:
14783: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14784: begin
14785: Const('ICMP_ECHO','LongInt').SetInt( 8);
14786: Const('ICMP_ECHOREPLY','LongInt').SetInt( 0);
14787: Const('ICMP_UNREACH','LongInt').SetInt( 3);
14788: Const('ICMP_TIME_EXCEEDED','LongInt').SetInt( 11);
14789: Const('ICMP6_ECHO','LongInt').SetInt( 128);
14790: Const('ICMP6_ECHOREPLY','LongInt').SetInt( 129);
14791: Const('ICMP6_UNREACH','LongInt').SetInt( 1);
14792: Const('ICMP6_TIME_EXCEEDED','LongInt').SetInt( 3);
14793: AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14794: + 'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14795: SIRegister_TPINGSend(CL);
14796: Function PingHost( const Host : string) : Integer';
14797: Function TraceRouteHost( const Host : string) : string';
14798: end;
14799:
14800: procedure SIRegister_asnutil(CL: TPSPascalCompiler);
14801: begin
14802: AddConstantN('synASN1_BOOL','LongWord').SetUInt( $01);
14803: Const('synASN1_INT','LongWord').SetUInt( $02);
14804: Const('synASN1_OCTSTR','LongWord').SetUInt( $04);
14805: Const('synASN1_NULL','LongWord').SetUInt( $05);
14806: Const('synASN1_OBJID','LongWord').SetUInt( $06);
14807: Const('synASN1_ENUM','LongWord').SetUInt( $0a);
14808: Const('synASN1_SEQ','LongWord').SetUInt( $30);
14809: Const('synASN1_SETOF','LongWord').SetUInt( $31);
14810: Const('synASN1_IPADDR','LongWord').SetUInt( $40);
14811: Const('synASN1_COUNTER','LongWord').SetUInt( $41);
14812: Const('synASN1_GAUGE','LongWord').SetUInt( $42);
14813: Const('synASN1_TIMETICKS','LongWord').SetUInt( $43);
14814: Const('synASN1_OPAQUE','LongWord').SetUInt( $44);
14815: Function synASNEncOIDItem( Value : Integer) : AnsiString';
14816: Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer';
14817: Function synASNEncLen( Len : Integer) : AnsiString';
14818: Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer';
14819: Function synASNEncInt( Value : Integer) : AnsiString';
14820: Function synASNEncUInt( Value : Integer) : AnsiString';
14821: Function synASNObject( const Data : AnsiString; ASNTYPE : Integer) : AnsiString';
14822: Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14823: Function synMibToId( Mib : String) : AnsiString';
14824: Function synIdToMib( const Id : AnsiString) : String';
14825: Function synIntMibToStr( const Value : AnsiString) : AnsiString';
14826: Function ASNDump( const Value : AnsiString) : AnsiString';
14827: Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean';
14828: Function LDAPResultDump( const Value : TLdapResultList) : AnsiString';
14829: end;
14830:
14831: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14832: begin
14833: Const('cLDAPProtocol','String').SetString( '389');
14834: Const('LDAP_ASN1_BIND_REQUEST','LongWord').SetUInt( $60);

```



```

14835: Const('LDAP_ASN1_BIND_RESPONSE','LongWord').SetUInt( $61);
14836: Const('LDAP_ASN1_UNBIND_REQUEST','LongWord').SetUInt( $42);
14837: Const('LDAP_ASN1_SEARCH_REQUEST','LongWord').SetUInt( $63);
14838: Const('LDAP_ASN1_SEARCH_ENTRY','LongWord').SetUInt( $64);
14839: Const('LDAP_ASN1_SEARCH_DONE','LongWord').SetUInt( $65);
14840: Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord').SetUInt( $73);
14841: Const('LDAP_ASN1_MODIFY_REQUEST','LongWord').SetUInt( $66);
14842: Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord').SetUInt( $67);
14843: Const('LDAP_ASN1_ADD_REQUEST','LongWord').SetUInt( $68);
14844: Const('LDAP_ASN1_ADD_RESPONSE','LongWord').SetUInt( $69);
14845: Const('LDAP_ASN1_DEL_REQUEST','LongWord').SetUInt( $4A);
14846: Const('LDAP_ASN1_DEL_RESPONSE','LongWord').SetUInt( $6B);
14847: Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord').SetUInt( $6C);
14848: Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord').SetUInt( $6D);
14849: Const('LDAP_ASN1_COMPARE_REQUEST','LongWord').SetUInt( $6E);
14850: Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord').SetUInt( $6F);
14851: Const('LDAP_ASN1_ABANDON_REQUEST','LongWord').SetUInt( $70);
14852: Const('LDAP_ASN1_EXT_REQUEST','LongWord').SetUInt( $77);
14853: Const('LDAP_ASN1_EXT_RESPONSE','LongWord').SetUInt( $78);
14854: SIRegister_TLDAPAttribute(CL);
14855: SIRegister_TLDAPAttributeList(CL);
14856: SIRegister_TLDAPResult(CL);
14857: SIRegister_TLDAPResultList(CL);
14858: AddTypeS('TLDAPModifyOp','( MO_Add, MO_Delete, MO_Replace )');
14859: AddTypeS('TLDAPSearchScope','( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14860: AddTypeS('TLDAPSearchAliases','( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14861: SIRegister_TLDAPSend(CL);
14862: Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString';
14863: end;
14864:
14865:
14866: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14867: begin
14868:   Const('cSysLogProtocol','String').SetString( '514');
14869:   Const('FCL_Kernel','LongInt').SetInt( 0);
14870:   Const('FCL_UserLevel','LongInt').SetInt( 1);
14871:   Const('FCL_MailSystem','LongInt').SetInt( 2);
14872:   Const('FCL_System','LongInt').SetInt( 3);
14873:   Const('FCL_Security','LongInt').SetInt( 4);
14874:   Const('FCL_Syslogd','LongInt').SetInt( 5);
14875:   Const('FCL_Printer','LongInt').SetInt( 6);
14876:   Const('FCL_News','LongInt').SetInt( 7);
14877:   Const('FCL_UUCP','LongInt').SetInt( 8);
14878:   Const('FCL_Clock','LongInt').SetInt( 9);
14879:   Const('FCL_Authorization','LongInt').SetInt( 10);
14880:   Const('FCL_FTP','LongInt').SetInt( 11);
14881:   Const('FCL_NTP','LongInt').SetInt( 12);
14882:   Const('FCL_LogAudit','LongInt').SetInt( 13);
14883:   Const('FCL_LogAlert','LongInt').SetInt( 14);
14884:   Const('FCL_Time','LongInt').SetInt( 15);
14885:   Const('FCL_Local0','LongInt').SetInt( 16);
14886:   Const('FCL_Local1','LongInt').SetInt( 17);
14887:   Const('FCL_Local2','LongInt').SetInt( 18);
14888:   Const('FCL_Local3','LongInt').SetInt( 19);
14889:   Const('FCL_Local4','LongInt').SetInt( 20);
14890:   Const('FCL_Local5','LongInt').SetInt( 21);
14891:   Const('FCL_Local6','LongInt').SetInt( 22);
14892:   Const('FCL_Local7','LongInt').SetInt( 23);
14893:   AddTypeS('TSyslogSeverity','( Emergency, Alert, Critical, Error, Warning,
14894:     +' Notice, Info, Debug )');
14895:   SIRegister_TSyslogMessage(CL);
14896:   SIRegister_TSyslogSend(CL);
14897:   Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
Content:string):Boolean;
14898: end;
14899:
14900:
14901: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14902: begin
14903:   AddTypeS('TMessPriority','( MP_unknown, MP_low, MP_normal, MP_high )');
14904:   SIRegister_TMessHeader(CL);
14905:   //AddTypeS('TMessHeaderClass','class of TMessHeader');
14906:   SIRegister_TMimeMess(CL);
14907: end;
14908:
14909: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14910: begin
14911:   (FindClass('TOBJECT'),'TmimePart');
14912:   AddTypeS('THookWalkPart','Procedure ( const Sender : TmimePart)');
14913:   AddTypeS('TmimePrimary','( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14914:   AddTypeS('TmimeEncoding','( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14915:   SIRegister_TMimePart(CL);
14916:   Const('MaxMimeType','LongInt').SetInt( 25);
14917:   Function GenerateBoundary : string';
14918: end;
14919:
14920: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14921: begin
14922:   Function InlineDecode( const Value : string; CP : TmimeChar ) : string';

```

```

14923: Function InlineEncode( const Value : string; CP, MimeP : TTimeChar ) : string';
14924: Function NeedInline( const Value : AnsiString ) : boolean';
14925: Function InlineCodeEx( const Value : string; FromCP : TTimeChar ) : string';
14926: Function InlineCode( const Value : string ) : string';
14927: Function InlineEmailEx( const Value : string; FromCP : TTimeChar ) : string';
14928: Function InlineEmail( const Value : string ) : string';
14929: end;
14930:
14931: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
14932: begin
14933:   Const('cFtpProtocol','String').SetString( '21');
14934:   Const('cFtpDataProtocol','String').SetString( '20');
14935:   Const('FTP_OK','LongInt').SetInt( 255);
14936:   Const('FTP_ERR','LongInt').SetInt( 254);
14937:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
14938:   SIRegister_TFTPListRec(CL);
14939:   SIRegister_TFTPList(CL);
14940:   SIRegister_TFTPSEND(CL);
14941:   Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean';
14942:   Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean';
14943:   Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
ToPort, ToFile, ToUser, ToPass : string ) : Boolean';
14944: end;
14945:
14946: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
14947: begin
14948:   Const('cHttpProtocol','String').SetString( '80');
14949:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
14950:   SIRegister_THTTSEND(CL);
14951:   Function HttpGetText( const URL : string; const Response : TStrings ) : Boolean';
14952:   Function HttpGetBinary( const URL : string; const Response : TStream ) : Boolean';
14953:   Function HttpPostBinary( const URL : string; const Data : TStream ) : Boolean';
14954:   Function HttpPostURL( const URL, URLData : string; const Data : TStream ) : Boolean';
14955:   Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
ResultData:TStrings):Boolean;
14956: end;
14957:
14958: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
14959: begin
14960:   Const('cSmtPProtocol','String').SetString( '25');
14961:   SIRegister_TSMTSEND(CL);
14962:   Function SendToRaw(const MailFrom,MailTo,SMTPHost:string;const MailData:TStrings;const Username,
Passw:string):Bool;
14963:   Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
14964:   Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
Username, Password : string):Boolean';
14965: end;
14966:
14967: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
14968: begin
14969:   Const('cSnmpProtocol','String').SetString( '161');
14970:   Const('cSnmpTrapProtocol','String').SetString( '162');
14971:   Const('SNMP_V1','LongInt').SetInt( 0);
14972:   Const('SNMP_V2C','LongInt').SetInt( 1);
14973:   Const('SNMP_V3','LongInt').SetInt( 3);
14974:   Const('PDUGetRequest','LongWord').SetUInt( $A0);
14975:   Const('PDUGetNextRequest','LongWord').SetUInt( $A1);
14976:   Const('PDUGetResponse','LongWord').SetUInt( $A2);
14977:   Const('PDUSetRequest','LongWord').SetUInt( $A3);
14978:   Const('PDUTrap','LongWord').SetUInt( $A4);
14979:   Const('PDUGetBulkRequest','LongWord').SetUInt( $A5);
14980:   Const('PDUInformRequest','LongWord').SetUInt( $A6);
14981:   Const('PDUTrapV2','LongWord').SetUInt( $A7);
14982:   Const('PDUReport','LongWord').SetUInt( $A8);
14983:   Const('ENoError','LongInt').SetInt( 0);
14984:   Const('ETooBig','LongInt').SetInt( 1);
14985:   Const('ENoSuchName','LongInt').SetInt( 2);
14986:   Const('EBadValue','LongInt').SetInt( 3);
14987:   Const('EReadOnly','LongInt').SetInt( 4);
14988:   Const('EGenErr','LongInt').SetInt( 5);
14989:   Const('ENoAccess','LongInt').SetInt( 6);
14990:   Const('EWrongType','LongInt').SetInt( 7);
14991:   Const('EWrongLength','LongInt').SetInt( 8);
14992:   Const('EWrongEncoding','LongInt').SetInt( 9);
14993:   Const('EWrongValue','LongInt').SetInt( 10);
14994:   Const('ENoCreation','LongInt').SetInt( 11);
14995:   Const('EInconsistentValue','LongInt').SetInt( 12);
14996:   Const('EResourceUnavailable','LongInt').SetInt( 13);
14997:   Const('ECommitFailed','LongInt').SetInt( 14);
14998:   Const('EUndoFailed','LongInt').SetInt( 15);
14999:   Const('EAuthorizationError','LongInt').SetInt( 16);
15000:   Const('ENotWritable','LongInt').SetInt( 17);
15001:   Const('EInconsistentName','LongInt').SetInt( 18);
15002:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15003:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15004:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15005:   SIRegister_TSNMPMib(CL);
15006:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer;
+EngineTime : integer; EngineStamp : Cardinal; end');
15007:

```

```

15008:   SIRegister_TSNMPRec(CL);
15009:   SIRegister_TSNMPSend(CL);
15010:   Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean'';
15011:   Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean'';
15012:   Function SNMPGetNext( var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15013:   Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15014:   Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):
Boolean;
15015:   Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer'';
15016:   Function RecvTrap( var Dest, Source, Enterprise, Community : AnsiString; var Generic, Specific, Seconds :
Integer; const MIBName, MIBValue : TStringList) : Integer'';
15017: end;
15018:
15019: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15020: begin
15021:   Function GetDomainName2: AnsiString'';
15022:   Function GetDomainController( Domain : AnsiString) : AnsiString'';
15023:   Function GetDomainUsers( Controller : AnsiString) : AnsiString'';
15024:   Function GetDomainGroups( Controller : AnsiString) : AnsiString'';
15025:   Function GetDateTime( Controller : AnsiString) : TDateTime'';
15026:   Function GetFullName2( Controller, UserID : AnsiString) : AnsiString'';
15027: end;
15028:
15029: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15030: begin
15031:   AddTypes('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15032:   AddTypes('TwwDateTimeSelection', '(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15033:   Function wwStrToDate( const S : string) : boolean'';
15034:   Function wwStrToTime( const S : string) : boolean'';
15035:   Function wwStrToDateTime( const S : string) : boolean'';
15036:   Function wwStrToTimeVal( const S : string) : TDateTime'';
15037:   Function wwStrToDateVal( const S : string) : TDateTime'';
15038:   Function wwStrToDateTimeVal( const S : string) : TDateTime'';
15039:   Function wwStrToInt( const S : string) : boolean'';
15040:   Function wwStrToFloat( const S : string) : boolean'';
15041:   Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder'';
15042:   Function wwNextDay( Year, Month, Day : Word) : integer'';
15043:   Function wwPriorDay( Year, Month, Day : Word) : integer'';
15044:   Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean'';
15045:   Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean'';
15046:   Function wwGetDateTimeCursorPosition( SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection'';
15047:   Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection'';
15048:   Function wwScanDate( const S : string; var Date : TDateTime) : Boolean'';
15049:   Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean'';
15050:   Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15051:   Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean'';
15052: end;
15053:
15054: unit uPSI_Themes;
15055: Function ThemeServices : TThemeServices'';
15056: Function ThemeControl( AControl : TControl) : Boolean'';
15057: Function UnthemedDesigner( AControl : TControl) : Boolean'';
15058: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15059: begin
15060:   Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String'';
15061: end;
15062: Unit uPSC_menus;
15063: Function StripHotkey( const Text : string) : string'';
15064: Function GetHotkey( const Text : string) : string'';
15065: Function AnsiSameCaption( const Text1, Text2 : string) : Boolean'';
15066: Function IsAltGRPressed : boolean'';
15067:
15068: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15069: begin
15070:   TCommandEvent', 'Procedure(Thread:TidPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15071:   SIRegister_TIdIMAP4Server(CL);
15072: end;
15073:
15074: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15075: begin
15076:   'HASH_SIZE','LongInt').SetInt( 256);
15077:   CL.FindClass('TOBJECT'),'EVariantSymbolTable'';
15078:   CL.AddTypes('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15079:   //CL.AddTypes('PSymbol', '^TSymbol // will not work'';
15080:   CL.AddTypes('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15081:   + ' : Integer; Value : Variant; end'';
15082:   //CL.AddTypes('PSymbolArray', '^TSymbolArray // will not work'';
15083:   SIRegister_TVariantSymbolTable(CL);
15084: end;
15085:
15086: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15087: begin
15088:   SIRegister_TThreadLocalVariables(CL);
15089:   Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar'';
15090:   //CL.AddDelphiFunction('Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD'';
15091:   Function ThreadLocals : TThreadLocalVariables'';
15092:   CL.AddDelphiFunction('Procedure WriteDebug( sz : String)'';
15093:   CL.AddConstantN('UDF_SUCCESS','LongInt').SetInt( 0);

```

```

15094: 'UDF_FAILURE','LongInt').SetInt( 1);
15095: 'cSignificantlyLarger','LongInt').SetInt( 1024 * 4);
15096: CL.AddTypeS('mTByteArray', 'array of byte;');
15097: function ChangeOEFFFromBytes(bFile:mTByteArray):Boolean;
15098: function ChangeOEFFFromFile(sFile:string; sDestFile:string):Boolean;
15099: procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15100: function IsNetworkConnected: Boolean;
15101: function IsInternetConnected: Boolean;
15102: function IsCOMConnected: Boolean;
15103: function IsNetworkOn: Boolean;
15104: function IsInternetOn: Boolean;
15105: function IsCOMOn: Boolean;
15106: function SetTimer( hWnd: HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc ): UINT);
15107: function KillTimer( hWnd: HWND; uIDEvent: UINT ): BOOL);
15108: function wIsWindowUnicode( hWnd: HWND ): BOOL);
15109: function wEnableWindow( hWnd: HWND; bEnable: BOOL ): BOOL);
15110: function wIsWindowEnabled( hWnd: HWND ): BOOL);
15111: function GetMenu( hWnd: HWND ): HMENU);
15112: function SetMenu( hWnd: HWND; hMenu: HMENU ): BOOL);
15113: end;
15114:
15115: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15116: begin
15117:   SIRegister_IDataBlock(CL);
15118:   SIRegister_ISendDataBlock(CL);
15119:   SIRegister_ITransport(CL);
15120:   SIRegister_TDataBlock(CL);
15121:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15122:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15123:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15124:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15125:   SIRegister_TCustomDataBlockInterpreter(CL);
15126:   SIRegister_TSendDataBlock(CL);
15127:   'CallSig','LongWord').SetUInt( $D800);
15128:   'ResultSig','LongWord').SetUInt( $D400);
15129:   'asMask','LongWord').SetUInt( $00FF);
15130:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15131:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15132:   Procedure CheckSignature( Sig: Integer);
15133: end;
15134:
15135: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15136: begin
15137:   CL.AddTypeS('HINTERNET', '__Pointer');
15138:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15139:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15140:   CL.AddTypeS('INTERNET_PORT', 'Word');
15141:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15142:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15143:   function InternetTimeFromSystemTime(const pst:TSystemTime; dwRFC:DWORD; lpzTime:LPSTR;
cbTime:DWORD):BOOL);
15144:   'INTERNET_RFC1123_FORMAT','LongInt').SetInt( 0);
15145:   'INTERNET_RFC1123_BUFSIZE','LongInt').SetInt( 30);
15146:   function InternetCrackUrl(lpzUrl:PChar; dwUrlLength,dwFlags:DWORD; var
lpUrlComponents:TURLComponents):BOOL;
15147:   function InternetCreateUrl( var lpUrlComponents: TURLComponents; dwFlags: DWORD; lpzUrl: PChar; var
dwUrlLength: DWORD ): BOOL);
15148:   function InternetDial( hwndParent: HWND; lpzConnectoid: PChar; dwFlags: DWORD; lpdwConnection:
DWORD; dwReserved: DWORD ): DWORD);
15149:   function InternetHangUp( dwConnection: DWORD; dwReserved: DWORD ): DWORD);
15150:   function InternetGoOnline( lpzURL: pchar; hwndParent: HWND; dwFlags: DWORD ): BOOL);
15151:   function InternetAutodial( dwFlags: DWORD; dwReserved: DWORD ): BOOL);
15152:   function InternetAutodialHangup( dwReserved: DWORD ): BOOL);
15153:   function InternetGetConnectedState( lpdwFlags: DWORD; dwReserved: DWORD ): BOOL);
15154:   function InternetCanonicalizeUrl(lpzUrl:PChar;lpzBuffer:PChar; var
lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL);
15155:   function InternetCombineUrl( lpzBaseUrl, lpzRelativeUrl: PChar; lpzBuffer: PChar; var
lpdwBufferLength: function InternetCloseHandle( hInet: HINTERNET ): BOOL);
15156:   function InternetConnect( hInet: HINTERNET; lpzServerName: PChar; nServerPort: INTERNET_PORT;
lpzUsername: PChar; lpzPassword: PChar; dwService: DWORD; dwFlags: DWORD; dwContext: DWORD ):
HINTERNET);
15157:   function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumberOfBytesAvailable:DWORD;dwFlags,
dwContext:DWORD): BOOL;
15158:   function WFTpGetFile( hConnect: HINTERNET; lpzRemoteFile: PChar; lpzNewFile: PChar; fFailIfExists:
BOOL; dwFlagsAndAttributes: DWORD; dwFlags: DWORD; dwContext: DWORD ): BOOL);
15159:   function WFTpPutFile( hConnect: HINTERNET; lpzLocalFile: PChar; lpzNewRemoteFile:PChar;dwFlags:DWORD;
dwContext: DWORD ): BOOL);
15160:   function FtpDeleteFile( hConnect: HINTERNET; lpzFileName: PChar ): BOOL);
15161:   function FtpRenameFile( hConnect: HINTERNET; lpzExisting: PChar; lpzNew: PChar ): BOOL);
15162:   function
FtpOpenFile(hConnect:HINTER;lpzFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15163:   function FtpCreateDirectory( hConnect: HINTERNET; lpzDirectory: PChar ): BOOL);
15164:   function FtpRemoveDirectory( hConnect: HINTERNET; lpzDirectory: PChar ): BOOL);
15165:   function FtpSetCurrentDirectory( hConnect: HINTERNET; lpzDirectory: PChar ): BOOL);
15166:   function FtpGetCurrentDirectory(hConnect:HINTER;lpzCurrentDir:PChar;var lpdwCurrentDirectory:DWORD):
BOOL;
15167:   function
FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpzCommand:PChar;dwContext:DWORD):BOOL;
15168:   function IS_GOPHER_FILE( GopherType: DWORD ): BOOL);

```



```

15169: Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15170: Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15171: Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15172: Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15173: Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15174: Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15175: Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15176: Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15177: Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15178: Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15179: Function
GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString :
PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15180: Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15181: Function
GopherOpenFile(hConnect:HINTERNET;lpszLocator:PChar;lpszView:PChar;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15182: Function HttpOpenRequest( hConnect : HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15183: Function
HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15184: Function HttpSendRequest(hRequest:HINTERNET; lpszHeaders : PChar; dwHeadersLength : DWORD; lpOptional :
TObject; dwOptionalLength:DWORD):BOOL;
15185: Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15186: Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15187: Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15188: Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15189: Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15190: Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15191: Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15192: Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
lpFirstCacheEntryInfo:TInternetCacheEntryInfo; var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15193: Function FindNextUrlCacheEntry( hEnumHandle : THandle; var lpNextCacheEntryInfo :
TInternetCacheEntryInfo; var lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15194: Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15195: Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15196: Function InternetDial(hWndParent:HWND; lpszConnectoid : Pchar; dwFlags : DWORD; lpdwConnection : DWORD;
dwReserved : DWORD) : DWORD');
15197: Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved : DWORD) : BOOL');
15198: end;
15199:
15200:
15201: Functions_max hex in the box maXbox
15202: functionslist.txt
15203: FunctionsList1 3.9.9.86/88/91/92
15204:
15205: *****
15206: Procedure
15207: PROCEDURE SIZE 7474 7401 6792 6310 5971 4438 3797 3600
15208: Procedure *****Now the Procedure list*****
15209: Procedure ( ACol, ARow : Integer; Items : TStrings)
15210: Procedure ( Agg : TAggregate)
15211: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
15212: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
15213: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
15214: Procedure ( ASender : TObject; const ABytes : Integer)
15215: Procedure ( ASender : TObject; VStream : TStream)
15216: Procedure ( AThread : TIdThread)
15217: Procedure ( AWebModule : TComponent)
15218: Procedure ( Column : TColumn)
15219: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
15220: Procedure ( const iStart : integer; const sText : string)
15221: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
15222: Procedure ( Database : TDatabase; LoginParams : TStrings)
15223: Procedure (DataSet:TCustomClientDataSet;E:EREconcileError;UpdateKind:TUpdateKind;var Action:
TReconcileAction)
15224: Procedure ( DATASET : TDATASET)
15225: Procedure ( DataSet:TDataset; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
15226: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
15227: Procedure ( DataSet : TDataset; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
15228: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
15229: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
15230: Procedure ( Done : Integer)
15231: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
15232: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
15233: Procedure
(HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
15234: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
15235: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
15236: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
15237: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
15238: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
15239: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TownerDrawState)
15240: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
15241: Procedure ( SENDER : TFIELD; const TEXT : String)
15242: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
15243: Procedure ( Sender : TIdTelnet; const Buffer : String)
15244: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
15245: Procedure ( SENDER : TObject; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
15246: Procedure ( SENDER : TObject; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)

```

```

15247: Procedure ( Sender : TObject; ACANVAS : TCanvas; var WIDTH, HEIGHT : Integer)
15248: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
15249: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
15250: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
15251: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
15252: Procedure ( Sender : TObject; Button : TMPBtnType)
15253: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
15254: Procedure ( Sender : TObject; Button : TUDBtnType)
15255: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
15256: Procedure ( Sender : TObject; ClientSocket : TServerClientWinSocket; var SocketThread : TServerClientThread)
15257: Procedure ( Sender : TObject; Column : TListColumn)
15258: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
15259: Procedure ( Sender : TObject; Connecting : Boolean)
15260: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
DoneDraw:Bool
15261: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
15262: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
15263: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
15264: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
15265: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
15266: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
15267: Procedure ( Sender : TObject; Index : Longint)
15268: Procedure ( Sender : TObject; Item : TListItem)
15269: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
15270: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
15271: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
15272: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
15273: Procedure ( Sender : TObject; Item : TListItem; var S : string)
15274: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
15275: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
15276: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
15277: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
15278: Procedure ( Sender : TObject; Node : TTreeNode)
15279: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
15280: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
15281: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
15282: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
15283: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
15284: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
15285: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
15286: Procedure ( Sender : TObject; Rect : TRect)
15287: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
15288: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
15289: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
15290: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
15291: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
15292: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
15293: Procedure ( SENDER : TObject; SOURCE : TMENUIITEM; REBUILD : BOOLEAN)
15294: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
15295: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
15296: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
15297: Procedure ( Sender : TObject; Thread : TServerClientThread)
15298: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
15299: Procedure ( Sender : TObject; Username, Password : string)
15300: Procedure ( Sender : TObject; var AllowChange : Boolean)
15301: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
15302: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
15303: Procedure ( Sender : TObject; var Continue : Boolean)
15304: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TidHTTPMethod)
15305: Procedure ( Sender : TObject; var Username : string)
15306: Procedure ( Sender : TObject; Wnd : HWND)
15307: Procedure ( Sender : TToolBar; Button : TToolButton)
15308: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
15309: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
15310: Procedure ( Sender : TToolBar; Index : Integer; var Allow : Boolean)
15311: Procedure ( Sender : TToolBar; Index : Integer; var Button : TToolButton)
15312: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusBarPanel; const Rect : TRect)
15313: Procedure ( StatusBar : TStatusBar; Panel : TStatusBarPanel; const Rect : TRect)
15314: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
15315: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
15316: procedure (Sender: TObject)
15317: procedure (Sender: TObject; var Done: Boolean)
15318: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
15319: procedure _T(Name: tbtString; v: Variant);
15320: Procedure AbandonSignalHandler( RtlSigNum : Integer)
15321: Procedure Abort
15322: Procedure About1Click( Sender : TObject)
15323: Procedure Accept( Socket : TSocket)
15324: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
15325: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
15326: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
15327: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
15328: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
15329: Procedure Add( Addend1, Addend2 : TMyBigInt)
15330: Procedure ADD( const AKEY, AVALUE : VARIANT)
15331: Procedure Add( const Key : string; Value : Integer)
15332: Procedure ADD( const NAME, FIELDS : string; OPTIONS : TINDEOPTIONS)
15333: Procedure ADD( FIELD : TFIELD)
15334: Procedure ADD( ITEM : TMENUIITEM)

```

```

15335: Procedure ADD( POPUP : TPOPUPMENU)
15336: Procedure AddCharacters( xCharacters : TCharSet)
15337: Procedure AddDriver( const Name : string; List : TStrings)
15338: Procedure AddImages( Value : TCustomImageList)
15339: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
15340: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
15341: Procedure AddLoader( Loader : TBitmapLoader)
15342: Procedure ADDPARAM( VALUE : TPARAM)
15343: Procedure AddPassword( const Password : string)
15344: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
15345: Procedure AddState( oState : TniRegularExpressionState)
15346: Procedure AddStrings( Strings : TStrings);
15347: procedure AddStrings(Strings: TStrings);
15348: Procedure AddStrings1( Strings : TWideStrings);
15349: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
15350: Procedure AddToRecentDocs( const Filename : string)
15351: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharSet)
15352: Procedure AllFunctionsList1Click( Sender : TObject)
15353: procedure AllObjectsList1Click(Sender: TObject);
15354: Procedure Allocate( AAllocateBytes : Integer)
15355: procedure AllResourceList1Click(Sender: TObject);
15356: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
15357: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
15358: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
15359: Procedure AnsiFree( var s : AnsiString)
15360: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
15361: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
15362: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
15363: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
15364: Procedure AntiFreeze;
15365: Procedure APPEND
15366: Procedure Append( const S : WideString)
15367: procedure Append(S: string);
15368: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
15369: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
15370: Procedure AppendChunk( Val : OleVariant)
15371: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
15372: Procedure AppendStr( var Dest : string; S : string)
15373: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
15374: Procedure ApplyRange
15375: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15376: Procedure Arrange( Code : TListArrangement)
15377: procedure Assert(expr : Boolean; const msg: string);
15378: procedure Assert2(expr : Boolean; const msg: string);
15379: Procedure Assign( AList : TCustomBucketList)
15380: Procedure Assign( Other : TObject)
15381: Procedure Assign( Source : TDragObject)
15382: Procedure Assign( Source : TPersistent)
15383: Procedure Assign(Source: TPersistent)
15384: procedure Assign2(mystring, mypath: string);
15385: Procedure AssignCurValues( Source : TDataSet);
15386: Procedure AssignCurValues1( const CurValues : Variant);
15387: Procedure ASSIGNFIELD( FIELD : TFIELD)
15388: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
15389: Procedure AssignFile(var F: Text; FileName: string)
15390: procedure AssignFile(var F: TextFile; FileName: string)
15391: procedure AssignFileRead(var mystring, myfilename: string);
15392: procedure AssignFileWrite(mystring, myfilename: string);
15393: Procedure AssignTo( Other : TObject)
15394: Procedure AssignValues( Value : TParameters)
15395: Procedure ASSIGNVALUES( VALUE : TPARAMS)
15396: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
15397: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
15398: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
15399: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
15400: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
15401: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
15402: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
15403: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
15404: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
15405: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
15406: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
15407: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
15408: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
15409: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
15410: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
15411: procedure Beep
15412: Procedure BeepOk
15413: Procedure BeepQuestion
15414: Procedure BeepHand
15415: Procedure BeepExclamation
15416: Procedure BeepAsterisk
15417: Procedure BeepInformation
15418: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
15419: Procedure BeginLayout
15420: Procedure BeginTimer( const Delay, Resolution : Cardinal)
15421: Procedure BeginUpdate
15422: procedure BeginUpdate;
15423: procedure BigScreen1Click(Sender: TObject);

```

```

15424: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
15425: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
15426: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
15427: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
15428: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
15429: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
15430: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
15431: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
15432: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
15433: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
15434: Procedure BreakPointMenuClick( Sender : TObject)
15435: procedure BRINGTOFRONT
15436: procedure BringToFront;
15437: Procedure btnBackClick( Sender : TObject)
15438: Procedure btnBrowseClick( Sender : TObject)
15439: Procedure BtnClick( Index : TNavigateBtn)
15440: Procedure btnLargeIconsClick( Sender : TObject)
15441: Procedure BuildAndSendRequest( AURI : TIdURI)
15442: Procedure BuildCache
15443: Procedure BurnMemory( var Buff, BuffLen : integer)
15444: Procedure BurnMemoryStream( Destructo : TMemoryStream)
15445: Procedure CalculateFirstSet
15446: Procedure Cancel
15447: procedure CancelDrag;
15448: Procedure CancelEdit
15449: procedure CANCELHINT
15450: Procedure CancelRange
15451: Procedure CancelUpdates
15452: Procedure CancelWriteBuffer
15453: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
15454: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
15455: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
15456: procedure CaptureScreenFormat(vname: string; vextension: string);
15457: procedure CaptureScreenPNG(vname: string);
15458: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
15459: procedure CASCADE
15460: Procedure CastNativeToSoap(Info:PTypInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
15461: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
15462: Procedure cbPathClick( Sender : TObject)
15463: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
15464: Procedure cedebugAfterExecute( Sender : TPSScript)
15465: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
15466: Procedure cedebugCompile( Sender : TPSScript)
15467: Procedure cedebugExecute( Sender : TPSScript)
15468: Procedure cedebugIdle( Sender : TObject)
15469: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
15470: Procedure CenterHeight( const pc, pcParent : TControl)
15471: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
15472: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
15473: Procedure Change
15474: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
15475: Procedure Changed
15476: Procedure ChangeDir( const ADirName : string)
15477: Procedure ChangeDirUp
15478: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
15479: Procedure ChangeLevelBy( Value : TChangeRange)
15480: Procedure ChDir(const s: string)
15481: Procedure Check(Status: Integer)
15482: Procedure CheckCommonControl( CC : Integer)
15483: Procedure CHECKFIELDNAME( const FIELDNAME : String)
15484: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
15485: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
15486: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
15487: Procedure CheckToken( T : Char)
15488: procedure CheckToken(t:char)
15489: Procedure CheckTokenSymbol( const S : string)
15490: procedure CheckTokenSymbol(s:string)
15491: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
15492: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15493: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
15494: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
15495: procedure CipherFile1Click(Sender: TObject);
15496: Procedure Clear;
15497: Procedure Clear1Click( Sender : TObject)
15498: Procedure ClearColor( Color : TColor)
15499: Procedure CLEARITEM( AITEM : TMENUITEM)
15500: Procedure ClearMapping
15501: Procedure ClearSelection( KeepPrimary : Boolean)
15502: Procedure ClearWriteBuffer
15503: Procedure Click
15504: Procedure Close
15505: Procedure Close1Click( Sender : TObject)
15506: Procedure CloseDatabase( Database : TDatabase)
15507: Procedure CloseDataSets
15508: Procedure CloseDialog
15509: Procedure CloseFile(var F: Text);
15510: Procedure Closure
15511: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
15512: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);

```



```

15513: Procedure CodeCompletionList1Click( Sender : TObject)
15514: Procedure ColEnter
15515: Procedure Collapse
15516: Procedure Collapse( Recurse : Boolean)
15517: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
15518: Procedure CommaSeparatedToStringList( AList : TStringList; const Value : string)
15519: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
15520: Procedure CompileClick( Sender : TObject)
15521: procedure ComponentCount1Click(Sender: TObject);
15522: Procedure Compress(azipfolder, azipfile: string)
15523: Procedure DeCompress(azipfolder, azipfile: string)
15524: Procedure XZip(azipfolder, azipfile: string)
15525: Procedure XUnZip(azipfolder, azipfile: string)
15526: Procedure Connect( const ATimeout : Integer)
15527: Procedure Connect( Socket : TSocket)
15528: procedure Console1Click(Sender: TObject);
15529: Procedure Continue
15530: Procedure ContinueCount( var Counter : TJclCounter)
15531: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
15532: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
15533: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
15534: Procedure ConvertImage(vsource, vdestination: string);
15535: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
15536: Procedure ConvertToGray(Cnv: TCanvas);
15537: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
15538: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
15539: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
15540: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
15541: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
15542: Procedure CopyFrom( mbCopy : TMyBigInt)
15543: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
15544: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
15545: Procedure CopyTidByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
of Byte; const ADestIndex : Integer; const ALength : Integer)
15546: Procedure CopyTidBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
ALen:Int)
15547: Procedure CopyTidCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
15548: Procedure CopyTidInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
15549: Procedure CopyTidIPv6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
15550: Procedure CopyTidLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
15551: Procedure CopyTidNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
15552: Procedure CopyTidNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
15553: Procedure CopyTidString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
15554: Procedure CopyTidWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
15555: Procedure CopyToClipboard
15556: Procedure CountParts
15557: Procedure CreateDataSet
15558: Procedure CreateEmptyFile( const FileName : string)
15559: Procedure CreateFileFromString( const FileName, Data : string)
15560: Procedure CreateFromDelta( Source : TPacketDataSet)
15561: procedure CREATEHANDLE
15562: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
15563: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
15564: Procedure CreateTable
15565: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
15566: procedure CSyntax1Click(Sender: TObject);
15567: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
15568: Procedure CURSORPOSCHANGED
15569: procedure CutFirstDirectory(var S: String)
15570: Procedure DataBaseError(const Message: string)
15571: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
15572: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
15573: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
15574: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
15575: Procedure DBIError(errorCode: Integer)
15576: Procedure DebugOutput( const AText : string)
15577: Procedure DebugRun1Click( Sender : TObject)
15578: procedure Dec;
15579: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
15580: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
15581: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
15582: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
15583: Procedure DecodeDateTime(const AValue:TDateTime;out AYear, AMonth, ADay, AHour, AMin, ASec, AMillSec:Word)
15584: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
15585: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear, AMonth, ANthDayOfWeek, ADayOfWeek:Word)
15586: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
15587: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
15588: Procedure DecompileClick( Sender : TObject)
15589: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
15590: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
15591: Procedure DeferLayout
15592: Procedure defFileRead
15593: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
15594: Procedure DelayMicroseconds( const MicroSeconds : Integer)
15595: Procedure Delete
15596: Procedure Delete( const AFilename : string)
15597: Procedure Delete( const Index : Integer)
15598: Procedure DELETE( INDEX : INTEGER)
15599: Procedure Delete( Index : LongInt)

```

```

15600: Procedure Delete( Node : TTreeNode)
15601: procedure Delete(var s: AnyString; ifrom, icount: Longint);
15602: Procedure DeleteAlias( const Name : string)
15603: Procedure DeleteDriver( const Name : string)
15604: Procedure DeleteIndex( const Name : string)
15605: Procedure DeleteKey( const Section, Ident : String)
15606: Procedure DeleteRecords
15607: Procedure DeleteRecords( AffectRecords : TAffectRecords)
15608: Procedure DeleteString( var pStr : String; const pDelStr : string)
15609: Procedure DeleteTable
15610: procedure DelphiSite1Click(Sender: TObject);
15611: Procedure Deselect
15612: Procedure Deselect( Node : TTreeNode)
15613: procedure DestroyComponents
15614: Procedure DestroyHandle
15615: Procedure Diff( var X : array of Double)
15616: procedure Diff(var X: array of Double);
15617: procedure DISABLEALIGN
15618: Procedure DisableConstraints
15619: Procedure Disconnect
15620: Procedure Disconnect( Socket : TSocket)
15621: Procedure Dispose
15622: procedure Dispose(P: PChar)
15623: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
15624: Procedure DoKey( Key : TDBCtrlGridKey)
15625: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
15626: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
15627: Procedure Dormant
15628: Procedure DoubleToBcd( const AValue : Double; var bcd : TBcd);
15629: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
15630: Procedure DoubleToComp( Value : Double; var Result : Comp)
15631: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
15632: procedure Draw(X, Y: Integer; Graphic: TGraphic);
15633: Procedure Draw1(Canvas:TCanvas; X,Y,
Index: Int; ADrawingStyle:TDrawingStyle; AImageType:TImageType; Enabled:Bool);
15634: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
15635: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
15636: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
15637: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
15638: procedure DrawFocusRect(const Rect: TRect);
15639: Procedure DrawHBIToTBitmap( HDIB : THandle; Bitmap : TBitmap)
15640: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
15641: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
15642: Procedure DrawOverlay1(Canvas:TCanvas; X,Y: Int; ImageIndex: Int; Overlay : TOverlay; ADrawingStyle :
TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
15643: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
15644: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
15645: Procedure DropConnections
15646: Procedure DropDown
15647: Procedure DumpDescription( oStrings : TStringList)
15648: Procedure DumpStateTable( oStrings : TStringList)
15649: Procedure EDIT
15650: Procedure EditButtonClick
15651: Procedure EditFont1Click( Sender : TObject)
15652: procedure Ellipse(X1, Y1, X2, Y2: Integer);
15653: Procedure Ellipse1( const Rect : TRect);
15654: Procedure EMMS
15655: Procedure Encode( ADest : TStream)
15656: procedure ENDDRAG(DROP:BOOLEAN)
15657: Procedure EndEdit( Cancel : Boolean)
15658: Procedure EndTimer
15659: Procedure EndUpdate
15660: Procedure EraseSection( const Section : string)
15661: Procedure Error( const Ident : string)
15662: procedure Error(Ident:Integer)
15663: Procedure ErrorFmt( const Ident : string; const Args : array of const)
15664: Procedure ErrorStr( const Message : string)
15665: procedure ErrorStr(Message:String)
15666: Procedure Exchange( Index1, Index2 : Integer)
15667: procedure Exchange(Index1, Index2: Integer);
15668: Procedure Exec( FileName, Parameters, Directory : string)
15669: Procedure ExecProc
15670: Procedure ExecSQL( UpdateKind : TUpdateKind)
15671: Procedure Execute
15672: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
15673: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
15674: Procedure ExecuteCommand(executeFile, paramstring: string)
15675: Procedure ExecuteShell(executeFile, paramstring: string)
15676: Procedure ExitThread(ExitCode: Integer); stdcall;
15677: Procedure ExitProcess(ExitCode: Integer); stdcall;
15678: Procedure Expand( AUserName : string; AResults : TStringList)
15679: Procedure Expand( Recurse : Boolean)
15680: Procedure ExportClipboard1Click( Sender : TObject)
15681: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
15682: Procedure ExtractContentFields( Strings : TStringList)
15683: Procedure ExtractCookieFields( Strings : TStringList)
15684: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStringList)
15685: Procedure ExtractHeaderFields(Separ,
WhiteSpace:TSysChSet;Cont:PChar;Strings:TStringList;Decode:Bool;StripQuots:Bool)

```

```

15686: Procedure ExtractHTTPFields(Separators,WhiteSpace:
      TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
15687: Procedure ExtractQueryFields( Strings : TStrings)
15688: Procedure FastDegToGrad
15689: Procedure FastDegToRad
15690: Procedure FastGradToDeg
15691: Procedure FastGradToRad
15692: Procedure FastRadToDeg
15693: Procedure FastRadToGrad
15694: Procedure FileClose( Handle : Integer)
15695: Procedure FileClose(handle: integer)
15696: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
15697: Procedure FileStructure( AStructure : TIdFTPDataStructure)
15698: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
15699: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
15700: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
15701: Procedure FillChar2(var X: PChar ; count: integer; value: char)
15702: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
15703: Procedure FillIPList
15704: procedure FillRect(const Rect: TRect);
15705: Procedure FillTStrings( AStrings : TStrings)
15706: Procedure FilterOnBookmarks( Bookmarks : array of const)
15707: procedure FinalizePackage(Module: HMODULE)
15708: procedure FindClose;
15709: procedure FindClose2(var F: TSearchRec)
15710: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
15711: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
15712: Procedure FindNearest( const KeyValues : array of const)
15713: Procedure FinishContext
15714: Procedure FIRST
15715: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
15716: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
15717: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
15718: Procedure FlushSchemaCache( const TableName : string)
15719: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
15720: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
15721: Procedure FormlClose( Sender : TObject; var Action : TCloseAction)
15722: Procedure FormActivate( Sender : TObject)
15723: procedure FormatLn(const format: String; const args: array of const); //alias
15724: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
15725: Procedure FormCreate( Sender : TObject)
15726: Procedure FormDestroy( Sender : TObject)
15727: Procedure FormKeyPress( Sender : TObject; var Key : Char)
15728: procedure FormOutput1Click(Sender: TObject);
15729: Procedure FormToHtml( Form : TForm; Path : string)
15730: procedure FrameRect(const Rect: TRect);
15731: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
15732: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
15733: Procedure Free( Buffer : TRecordBuffer)
15734: Procedure Free( Buffer : TValueBuffer)
15735: Procedure Free;
15736: Procedure FreeAndNil(var Obj:TObject)
15737: Procedure FreeImage
15738: procedure FreeMem(P: PChar; Size: Integer)
15739: Procedure FreeTreeData( Tree : TUpdateTree)
15740: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
15741: Procedure FullCollapse
15742: Procedure FullExpand
15743: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
15744: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
15745: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
15746: Procedure Get1( AURL : string; const AResponseContent : TStream);
15747: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
15748: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
15749: Procedure GetAliasNames( List : TStrings)
15750: Procedure GetAliasParams( const AliasName : string; List : TStrings)
15751: Procedure GetApplicationsRunning( Strings : TStrings)
15752: Procedure GetCommandTypes( List : TWideStrings)
15753: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
15754: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
15755: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
15756: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
15757: Procedure GetDatabaseNames( List : TStrings)
15758: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
15759: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
15760: Procedure GetDir(d: byte; var s: string)
15761: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
15762: Procedure GetDriverNames( List : TStrings)
15763: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
15764: Procedure GetDriverParams( const DriverName : string; List : TStrings)
15765: Procedure GetEMails1Click( Sender : TObject)
15766: Procedure getEnvironmentInfo;
15767: Function getEnvironmentString: string;
15768: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
15769: Procedure GetFieldNames( const TableName : string; List : TStrings)
15770: Procedure GetFieldNames( const TableName : string; List : TStrings);
15771: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
15772: Procedure GETFIELDNAMES( LIST : TSTRINGS)
15773: Procedure GetFieldNames1( const TableName : string; List : TStrings);

```

```

15774: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStringList);
15775: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings);
15776: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
15777: Procedure GetFileAttributeList( const Items : TStringList; const Attr : Integer);
15778: Procedure GetFileAttributeListEx( const Items : TStringList; const Attr : Integer);
15779: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd);
15780: Procedure GetFormatSettings;
15781: Procedure GetFromDIB( var DIB : TBitmapInfo);
15782: Procedure GetFromHDIB( HDIB : HBitmap);
15783: Procedure GetIcon( Index : Integer; Image : TIcon);
15784: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
15785: Procedure GetIndexInfo( IndexName : string);
15786: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStringList);
15787: Procedure GetIndexNames( List : TStringList);
15788: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
15789: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
15790: Procedure GetIndexNames4( const TableName : string; List : TStringList);
15791: Procedure GetInternalResponse;
15792: Procedure GETITEMNAMES( LIST : TStrings);
15793: procedure GetMem(P: PChar; Size: Integer);
15794: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER);
15795: Procedure GetPackageDescription(ModuleName: PChar): string);
15796: Procedure GetPackageNames( List : TStringList);
15797: Procedure GetPackageNames1( List : TWideStrings);
15798: Procedure GetParamList( List : TList; const ParamNames : WideString);
15799: Procedure GetProcedureNames( List : TStringList);
15800: Procedure GetProcedureNames( List : TWideStrings);
15801: Procedure GetProcedureNames1( const PackageName : string; List : TStringList);
15802: Procedure GetProcedureNames1( List : TStringList);
15803: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStringList);
15804: Procedure GetProcedureNames3( List : TWideStrings);
15805: Procedure GetProcedureNames4( const PackageName : WideString; List : TWideStrings);
15806: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
15807: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
15808: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
15809: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : WideString; List : TList);
15810: Procedure GetProviderNames( Names : TWideStrings);
15811: Procedure GetProviderNames( Proc : TGetStrProc);
15812: Procedure GetProviderNames1( Names : TStringList);
15813: procedure GetQRCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
15814: procedure GetQRCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no autoopen
image
15815: Function GetQRCode4(Width,Height:Word;Correct_Level:string; const
Data:string;aformat:string):TLinearBitmap;
15816: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte);
15817: Procedure GetSchemaNames( List : TStringList);
15818: Procedure GetSchemaNames1( List : TWideStrings);
15819: Procedure GetSessionNames( List : TStringList);
15820: Procedure GetStoredProcNames( const DatabaseName : string; List : TStringList);
15821: Procedure GetStrings( List : TStringList);
15822: Procedure GetSystemTime; stdcall;
15823: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings);
15824: Procedure GetTableNames( List : TStringList; SystemTables : Boolean);
15825: Procedure GetTableNames( List : TStringList; SystemTables : Boolean);
15826: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
15827: Procedure GetTableNames1( List : TStringList; SchemaName : WideString; SystemTables : Boolean);
15828: Procedure GetTableNames1( List : TStringList; SystemTables : Boolean);
15829: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
15830: Procedure GetTransitionsOn( cChar : char; oStateList : TList);
15831: Procedure GetVisibleWindows( List : TStringList);
15832: Procedure GoBegin;
15833: Procedure GotoCurrent( DataSet : TCustomClientDataSet);
15834: Procedure GotoCurrent( Table : TTable);
15835: procedure GotoEnd1Click(Sender: TObject);
15836: Procedure GotoNearest;
15837: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
Direction: TGradientDirection);
15838: Procedure HandleException( E : Exception; var Handled : Boolean);
15839: procedure HANDLEMESSAGE;
15840: procedure HandleNeeded;
15841: Procedure Head( AURL : string);
15842: Procedure Help( var AHelpContents : TStringList; ACommand : string);
15843: Procedure HexToBinary( Stream : TStream);
15844: procedure HexToBinary(Stream:TStream);
15845: Procedure HideDragImage;
15846: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean);
15847: Procedure HideTraybar;
15848: Procedure HideWindowForSeconds(secs: integer); { //3 seconds}
15849: Procedure HideWindowForSeconds2(secs: integer; apphandle, aSelf: TForm); { //3 seconds}
15850: Procedure HookOSExceptions;
15851: Procedure HookSignal( RtlSigNum : Integer);
15852: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
15853: Procedure HTMLSyntax1Click( Sender : TObject);
15854: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler);
15855: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter);
15856: Procedure ImportFromClipboard1Click( Sender : TObject);
15857: Procedure ImportFromClipboard2Click( Sender : TObject);
15858: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer);
15859: procedure Incb(var x: byte);

```



```

15860: Procedure Include1Click( Sender : TObject)
15861: Procedure IncludeOFF; //preprocessing
15862: Procedure IncludeON;
15863: procedure InfolClick(Sender: TObject);
15864: Procedure InitAltRecBuffers( CheckModified : Boolean)
15865: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
15866: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
15867: Procedure InitData( ASource : TDataSet)
15868: Procedure InitDelta( ADelta : TPacketDataSet);
15869: Procedure InitDeltal( const ADelta : OleVariant);
15870: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
15871: Procedure Initialize
15872: procedure InitializePackage(Module: HMODULE)
15873: Procedure INITIATEACTION
15874: Procedure initHexArray(var hexn: THexArray); //THexArray', 'array[0..15] of char;
15875: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
15876: Procedure InitModule( AModule : TComponent)
15877: Procedure InitStdConvs
15878: Procedure InitTreeData( Tree : TUpdateTree)
15879: Procedure INSERT
15880: Procedure Insert( Index : Integer; AClass : TClass)
15881: Procedure Insert( Index : Integer; AComponent : TComponent)
15882: Procedure Insert( Index : Integer; AObject : TObject)
15883: Procedure Insert( Index : Integer; const S : WideString)
15884: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
15885: Procedure Insert(Index: Integer; const S: string);
15886: procedure Insert(Index: Integer; S: string);
15887: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
15888: procedure InsertComponent(AComponent:TComponent)
15889: procedure InsertControl(AControl: TControl);
15890: Procedure InsertIcon( Index : Integer; Image : TIcon)
15891: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
15892: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
15893: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
15894: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
15895: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
15896: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
15897: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
15898: Procedure InternalBeforeResolve( Tree : TUpdateTree)
15899: Procedure InvalidateModuleCache
15900: Procedure InvalidateTitles
15901: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
15902: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
15903: Procedure InvalidDateTimeError(const AYear, AMth, ADay, AHour, AMin, ASec, AMilSec: Word; const
ABaseDate:TDateTime)
15904: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
15905: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
15906: procedure JavaSyntax1Click(Sender: TObject);
15907: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
15908: Procedure KillDataChannel
15909: Procedure Largefont1Click( Sender : TObject)
15910: Procedure LAST
15911: Procedure LaunchCpl( FileName : string)
15912: Procedure Launch( const AFile : string)
15913: Procedure LaunchFile( const AFile : string)
15914: Procedure LetFileList(FileList: TStringlist; apath: string);
15915: Procedure lineToNumber( xmemo : String; met : boolean)
15916: Procedure ListViewCustomDrawItem(Sender:TCustomListView;Item:TListItem;State:TCustDrawState;var
DefaultDraw:Bool)
15917: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
: TCustomDrawState; var DefaultDraw : Boolean)
15918: Procedure ListViewData( Sender : TObject; Item : TListItem)
15919: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
: TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
15920: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
15921: Procedure ListViewDblClick( Sender : TObject)
15922: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
15923: Procedure ListDLLEExports(const FileName: string; List: TStrings);
15924: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
15925: procedure LoadBytecode1Click(Sender: TObject);
15926: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
15927: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
15928: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
15929: Procedure LoadFromFile( AFileName : string)
15930: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
15931: Procedure LoadFromFile( const FileName : string)
15932: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
15933: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
15934: Procedure LoadFromFile( const FileName : WideString)
15935: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
15936: Procedure LoadFromFile(const AFileName: string)
15937: procedure LoadFromFile(FileName: string);
15938: procedure LoadFromFile(FileName:String)
15939: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
15940: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
15941: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
15942: Procedure LoadFromStream( const Stream : TStream)
15943: Procedure LoadFromStream( S : TStream)
15944: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)

```

```

15945: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
15946: Procedure LoadFromStream( Stream : TStream)
15947: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
15948: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
15949: procedure LoadFromStream(Stream: TStream);
15950: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
15951: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
15952: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
15953: Procedure LoadLastFile1Click( Sender : TObject)
15954: { LoadIcoToImage loads two icons from resource named NameRes,
15955:   into two image lists ALarge and ASmall}
15956: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
15957: Procedure LoadMemo
15958: Procedure LoadParamsFromIniFile( FFileName : WideString)
15959: Procedure Lock
15960: Procedure Login
15961: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
15962: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
15963: Procedure MakeCaseInsensitive
15964: Procedure MakeDeterministic( var bChanged : boolean)
15965: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
15966: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
15967: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
15968: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
15969: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
15970: Procedure SetRectComplexFormatStr( const S : string)
15971: Procedure SetPolarComplexFormatStr( const S : string)
15972: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
15973: Procedure MakeVisible
15974: Procedure MakeVisible( PartialOK : Boolean)
15975: Procedure ManuallClick( Sender : TObject)
15976: Procedure MarkReachable
15977: Procedure maXbox; //shows the exe version data in a win box
15978: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
15979: Procedure Memo1Change( Sender : TObject)
15980: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
  Action:TSynReplaceAction)
15981: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
15982: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
15983: procedure Memory1Click(Sender: TObject);
15984: Procedure MERGE( MENU : TMAINMENU)
15985: Procedure MergeChangeLog
15986: procedure MINIMIZE
15987: Procedure MinimizeMaxbox;
15988: Procedure MkDir(const s: string)
15989: Procedure mnuPrintFont1Click( Sender : TObject)
15990: procedure ModalStarted
15991: Procedure Modified
15992: Procedure ModifyAlias( Name : string; List : TStrings)
15993: Procedure ModifyDriver( Name : string; List : TStrings)
15994: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
15995: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
15996: Procedure Move( CurIndex, NewIndex : Integer)
15997: procedure Move(CurIndex, NewIndex: Integer);
15998: Procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
15999: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
16000: Procedure moveCube( o : TMyLabel)
16001: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
16002: procedure MoveTo(X, Y: Integer);
16003: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
16004: Procedure MovePoint(var x,y:Extended; const angle:Extended);
16005: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
16006: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
16007: Procedure MsgAbout( Handle: Int; const Msg, Caption: string; const IcoName: string = 'MAINICON'; Flags: DWORD=MB_OK);
16008: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
16009: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
16010: procedure New(P: PChar)
16011: procedure New1Click(Sender: TObject);
16012: procedure NewInstancelClick(Sender: TObject);
16013: Procedure NEXT
16014: Procedure NextMonth
16015: Procedure Noop
16016: Procedure NormalizePath( var APath : string)
16017: procedure ObjectBinaryToText(Input, Output: TStream)
16018: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16019: procedure ObjectResourceToText(Input, Output: TStream)
16020: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16021: procedure ObjectTextToBinary(Input, Output: TStream)
16022: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16023: procedure ObjectTextToResource(Input, Output: TStream)
16024: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16025: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
16026: Procedure Open( const UserID : WideString; const Password : WideString);
16027: Procedure Open;
16028: Procedure open1Click( Sender : TObject)
16029: Procedure OpenCdDrive
16030: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
16031: Procedure OpenCurrent
16032: Procedure OpenFile(vfilenamepath: string)

```

```

16033: Procedure OpenDirectory1Click( Sender : TObject)
16034: Procedure OpenIndexFile( const IndexName : string)
16035: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
SchemaID:OleVariant;DataSet:TADODataset)
16036: Procedure OpenWriteBuffer( const AThreshold : Integer)
16037: Procedure OptimizeMem
16038: Procedure Options1( AURL : string);
16039: Procedure OutputDebugString(lpOutputString : PChar)
16040: Procedure PackBuffer
16041: Procedure Paint
16042: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
16043: Procedure PaintToTBitmap( Target : TBitmap)
16044: Procedure PaletteChanged
16045: Procedure ParentBiDiModeChanged
16046: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
16047: Procedure PasteFromClipboard;
16048: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
16049: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
16050: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
16051: Procedure PError( Text : string)
16052: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
16053: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
16054: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
16055: procedure playmp3(mpath: string);
16056: Procedure PlayMP3Click( Sender : TObject)
16057: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
16058: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
16059: procedure PolyBezier(const Points: array of TPoint);
16060: procedure PolyBezierTo(const Points: array of TPoint);
16061: procedure Polygon(const Points: array of TPoint);
16062: procedure Polyline(const Points: array of TPoint);
16063: Procedure Pop
16064: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
16065: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
16066: Procedure POPUP( X, Y : INTEGER)
16067: Procedure PopupURL(URL : WideString);
16068: Procedure POST
16069: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
16070: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
16071: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
16072: Procedure PostUser( const Email, FirstName, LastName : WideString)
16073: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
16074: procedure Pred(X: int64);
16075: Procedure Prepare
16076: Procedure PrepareStatement
16077: Procedure PreProcessXML( AList : TStrings)
16078: Procedure PreventDestruction
16079: Procedure Print( const Caption : string)
16080: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
16081: procedure printf(const format: String; const args: array of const);
16082: Procedure PrintList(Value: TStringList);
16083: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
16084: Procedure Printout1Click( Sender : TObject)
16085: Procedure ProcessHeaders
16086: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
16087: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
16088: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
16089: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
16090: Procedure ProcessMessagesOFF; //application.processmessages
16091: Procedure ProcessMessagesON;
16092: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
16093: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
16094: Procedure Proclst Size is: 3797 /1415
16095: Procedure procMessClick( Sender : TObject)
16096: Procedure PSScriptCompile( Sender : TPSScript)
16097: Procedure PSScriptExecute( Sender : TPSScript)
16098: Procedure PSScriptLine( Sender : TObject)
16099: Procedure Push( ABoundary : string)
16100: procedure PushItem(AItem: Pointer)
16101: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
16102: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
16103: procedure PutLinuxLines(const Value: string)
16104: Procedure Quit
16105: Procedure RaiseConversionError( const AText : string);
16106: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
16107: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
16108: procedure RaiseException(Ex: TIFException; Param: String);
16109: Procedure RaiseExceptionForLastCmdResult;
16110: procedure RaiseLastException;
16111: procedure RaiseException2;
16112: Procedure RaiseLastOSError
16113: Procedure RaiseLastWin32;
16114: procedure RaiseLastWin32Error)
16115: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
16116: Procedure RandomFillStream( Stream : TMemoryStream)
16117: procedure randomize;
16118: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
16119: Procedure RCS
16120: Procedure Read( Socket : TSocket)

```

```

16121: Procedure ReadBlobData
16122: procedure ReadBuffer(Buffer:String;Count:LongInt)
16123: procedure ReadOnly1Click(Sender: TObject);
16124: Procedure ReadSection( const Section : string; Strings : TStringList)
16125: Procedure ReadSections( Strings : TStringList)
16126: Procedure ReadSections( Strings : TStringList);
16127: Procedure ReadSections1( const Section : string; Strings : TStringList);
16128: Procedure ReadSectionValues( const Section : string; Strings : TStringList)
16129: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
16130: Procedure ReadStrings( ADest : TStringList; AReadLinesCount : Integer)
16131: Procedure ReadVersion2(aFileName: STRING; aVersion : TStringList);
16132: Function ReadVersion(aFileName: STRING; aVersion : TStringList): boolean;
16133: Procedure Realign;
16134: procedure Rectangle(X1, Y1, X2, Y2: Integer);
16135: Procedure Rectangle1( const Rect : TRect);
16136: Procedure RectCopy( var Dest : TRect; const Source : TRect)
16137: Procedure RectFitToScreen( var R : TRect)
16138: Procedure RectGrow( var R : TRect; const Delta : Integer)
16139: Procedure RectGrowX( var R : TRect; const Delta : Integer)
16140: Procedure RectGrowY( var R : TRect; const Delta : Integer)
16141: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
16142: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
16143: Procedure RectNormalize( var R : TRect)
16144: // TFileCallbackProcedure = procedure(filename:string);
16145: Procedure RecurseDirectory(Dir: String; IncludeSubs: boolean; callback: TFileCallbackProcedure);
16146: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
16147: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
16148: Refresh;
16149: Procedure RefreshData( Options : TFetchOptions)
16150: Procedure REFRESHLOOKUPLIST
16151: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
16152: Procedure RegisterChanges( Value : TChangeLink)
16153: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
16154: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
16155: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
16156: Procedure ReInitialize( ADelay : Cardinal)
16157: procedure RELEASE
16158: Procedure Remove( const AByteCount : integer)
16159: Procedure REMOVE( FIELD : TFIELD)
16160: Procedure REMOVE( ITEM : TMENUITEM)
16161: Procedure REMOVE( POPUP : TPOPMENU)
16162: Procedure RemoveAllPasswords
16163: procedure RemoveComponent(AComponent:TComponent)
16164: Procedure RemoveDir( const ADirName : string)
16165: Procedure RemoveLambdaTransitions( var bChanged : boolean)
16166: Procedure REMOVEPARAM( VALUE : TPARAM)
16167: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
16168: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
16169: Procedure Rename( const ASourceFile, ADestFile : string)
16170: Procedure Rename( const FileName : string; Reload : Boolean)
16171: Procedure RenameTable( const NewTableName : string)
16172: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
16173: Procedure Replace1Click( Sender : TObject)
16174: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
16175: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
16176: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
16177: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
16178: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
16179: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
16180: Procedure Requery( Options : TExecuteOptions)
16181: Procedure Reset
16182: Procedure Reset1Click( Sender : TObject)
16183: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
16184: procedure ResourceExplore1Click(Sender: TObject);
16185: Procedure RestoreContents
16186: Procedure RestoreDefaults
16187: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
16188: Procedure RetrieveHeaders
16189: Procedure RevertRecord
16190: Procedure RGBAToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
16191: Procedure RGBAToBGR1( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16192: Procedure RGBAToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16193: Procedure RGBAToHSL( const R, G, B : Single; out H, S, L : Single);
16194: Procedure RGBAToHSL1( const RGB : TColor32; out H, S, L : Single);
16195: Procedure RGBAToHSV( r, g, b : Integer; var h, s, v : Integer)
16196: Procedure RleCompress2( Stream : TStream)
16197: Procedure RleDecompress2( Stream : TStream)
16198: Procedure Rmdir(const S: string)
16199: Procedure Rollback
16200: Procedure Rollback( TransDesc : TTransactionDesc)
16201: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
16202: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
16203: Procedure RollbackTrans
16204: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
16205: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
16206: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
16207: Procedure RunDll32Internal( Wnd : HWND; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
16208: Procedure S_AddMessageToStrings( AMessages : TStringList; AMsg : string)
16209: Procedure S_EBox( const AText : string)

```



```

16210: Procedure S_GetEncryptionKeys(DateTImel,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
AddKey:int
16211: Procedure S_IBox( const AText : string)
16212: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
16213: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
16214: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
16215: Procedure SampleVarianceAndMean
16216: ( const X : TDynFloatArray; var Variance, Mean : Float)
16217: Procedure Save2Click( Sender : TObject)
16218: Procedure Saveas3Click( Sender : TObject)
16219: Procedure Savebefore1Click( Sender : TObject)
16220: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
16221: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
16222: Procedure SaveConfigFile
16223: Procedure SaveOutput1Click( Sender : TObject)
16224: procedure SaveScreenshotClick(Sender: TObject);
16225: Procedure SaveLn(pathname, content: string); //SaveLn(exepath+'mysavelntest.txt', memo2.text);
16226: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APALETTE : HPALETTE)
16227: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APALETTE : HPALETTE)
16228: Procedure SaveToFile( AFileName : string)
16229: Procedure SAVETOFILE( const FILENAME : String)
16230: Procedure SaveToFile( const FileName : WideString)
16231: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
16232: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
16233: procedure SaveToFile(FileName: string);
16234: procedure SaveToFile(FileName:String)
16235: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
16236: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16237: Procedure SaveToStream( S : TStream)
16238: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16239: Procedure SaveToStream( Stream : TStream)
16240: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
16241: procedure SaveToStream(Stream:TStream);
16242: procedure SaveToStream(Stream:TStream)
16243: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
16244: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
16245: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
16246: procedure Say(const sText: string)
16247: Procedure SBytecode1Click( Sender : TObject)
16248: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
16249: procedure ScriptExplorer1Click(Sender: TObject);
16250: Procedure Scroll( Distance : Integer)
16251: Procedure Scroll( DX, DY : Integer)
16252: procedure ScrollBy(DeltaX, DeltaY: Integer);
16253: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
16254: Procedure ScrollTabs( Delta : Integer)
16255: Procedure Search1Click( Sender : TObject)
16256: procedure SearchAndOpenDoc(vfilenamepath: string)
16257: procedure SearchAndOpenFile(vfilenamepath: string)
16258: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
16259: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
16260: Procedure SearchNext1Click( Sender : TObject)
16261: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
16262: Procedure Select1( const Nodes : array of TTreeNode);
16263: Procedure Select2( Nodes : TList);
16264: Procedure SelectNext( Direction : Boolean)
16265: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
16266: Procedure SelfTestPEM //unit uPSI_cPEM
16267: Procedure Send( AMsg : TIdMessage)
16268: //config forst in const MAILINIFILE = 'maildef.ini';
16269: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox',
16270: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
16271: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
16272: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
16273: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
16274: Procedure SendResponse
16275: Procedure SendStream( AStream : TStream)
16276: Procedure Set8087CW( NewCW : Word)
16277: Procedure SetAll( One, Two, Three, Four : Byte)
16278: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
16279: Procedure SetAppDispatcher( const ADispatcher : TComponent)
16280: procedure SetArrayLength;
16281: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
16282: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
16283: Procedure SetAsHandle( Format : Word; Value : THandle)
16284: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
16285: procedure SetCaptureControl(Control: TControl);
16286: Procedure SetColumnAttributes
16287: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
16288: Procedure SetCustomHeader( const Name, Value : string)
16289: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
FieldName:Widestring)
16290: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
16291: Procedure SetFocus
16292: procedure SetFocus; virtual;
16293: Procedure SetInitialState
16294: Procedure SetKey
16295: procedure SetLastError(ErrorCode: Integer)
16296: procedure SetLength;

```

```

16297: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
16298: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
16299: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
16300: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
16301: Procedure SetParams1( UpdateKind : TUpdateKind);
16302: Procedure SetPassword( const Password : string)
16303: Procedure SetPointer( Ptr : Pointer; Size : Longint)
16304: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
16305: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
16306: Procedure SetProvider( Provider : TComponent)
16307: Procedure SetProxy( const Proxy : string)
16308: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
16309: Procedure SetRange( const StartValues, EndValues : array of const)
16310: Procedure SetRangeEnd
16311: Procedure SetRate( const aPercent, aYear : integer)
16312: procedure SetRate(const aPercent, aYear: integer)
16313: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
16314: Procedure SetSafeCallExceptionMsg( Msg : string)
16315: procedure SETSELTEXTBUF(BUFFER:PCHAR)
16316: Procedure SetSize( AWidth, AHeight : Integer)
16317: procedure SetSize(NewSize:LongInt)
16318: procedure SetString(var s: string; buffer: PChar; len: Integer)
16319: Procedure SetStrings( List : TStrings)
16320: Procedure SetText( Text : PwideChar)
16321: procedure SetText(Text: PChar);
16322: Procedure SetTextBuf( Buffer : PChar)
16323: procedure SETTEXTBUF(BUFFER:PCHAR)
16324: Procedure SetTick( Value : Integer)
16325: Procedure SetTimeout( ATimeout : Integer)
16326: Procedure SetTraceEvent( Event : TDBXTraceEvent)
16327: Procedure SetUserName( const UserName : string)
16328: Procedure SetWallpaper( Path : string);
16329: procedure ShellStyle1Click(Sender: TObject);
16330: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
16331: Procedure ShowFileProperties( const FileName : string)
16332: Procedure ShowIncludelClick( Sender : TObject)
16333: Procedure ShowInterfaces1Click( Sender : TObject)
16334: Procedure ShowLastException1Click( Sender : TObject)
16335: Procedure ShowMessage( const Msg : string)
16336: Procedure ShowMessageBig(const aText : string);
16337: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
16338: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
16339: Procedure MsgBig(const aText : string); //alias
16340: procedure showmessage(mytext: string);
16341: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
16342: procedure ShowMessageFmt(const Msg: string; Params: array of const))
16343: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
16344: procedure ShowMessageBig(const Msg: string; X: Integer; Y: Integer))
16345: Procedure ShowSearchDialog( const Directory : string)
16346: Procedure ShowSpecChars1Click( Sender : TObject)
16347: Procedure ShowBitmap(bmap: TBitmap); //draw in a form!
16348: Procedure ShredFile( const FileName : string; Times : Integer)
16349: procedure Shuffle(vQ: TStringList);
16350: Procedure ShuffleList( var List : array of Integer; Count : Integer)
16351: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
16352: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
16353: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
16354: Procedure Site( const ACommand : string)
16355: Procedure SkipEOL
16356: Procedure Sleep( ATime : cardinal)
16357: Procedure Sleep( milliseconds : Cardinal)
16358: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
16359: Procedure Slinenumber1Click( Sender : TObject)
16360: Procedure Sort
16361: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
16362: procedure Speak(const sText: string) //async like voice
16363: procedure Speak2(const sText: string) //sync
16364: procedure Split(Str: string; SubStr: string; List: TStrings);
16365: Procedure SplitNameValue( const Line : string; var Name, Value : string)
16366: Procedure SplitColumns( const AData : string; AStrings : TStrings; const ADelim : string)
16367: Procedure SplitColumnsNoTrim( const AData : string; AStrings : TStrings; const ADelim : string)
16368: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
16369: Procedure SplitString( const AStr, AToken : string; var VLeft, VRight : string)
16370: procedure SQLSyntax1Click(Sender: TObject);
16371: Procedure SRand( Seed : RNG_IntType)
16372: Procedure Start
16373: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
16374: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
16375: Procedure StartTransaction( TransDesc : TTransactionDesc)
16376: Procedure Status( var AStatusList : TStringList)
16377: Procedure StatusBar1Db1Click( Sender : TObject)
16378: Procedure StepInto1Click( Sender : TObject)
16379: Procedure StepIt
16380: Procedure StepOut1Click( Sender : TObject)
16381: Procedure Stop
16382: procedure stopmp3;
16383: procedure StartWeb(aurl: string);
16384: Str(aint: integer; astr: string); //of system
16385: Procedure StrDispose( Str : PChar)

```

```

16386: procedure StrDispose(Str: PChar)
16387: Procedure StrReplace(var Str: String; Old, New: String);
16388: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
16389: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
16390: Procedure StringToBytes( Value : String; Bytes : array of byte)
16391: procedure StrSet(c : Char; I : Integer; var s : String);
16392: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
16393: Procedure StructureMount( APath : String)
16394: procedure STYLECHANGED(SENDER:TObject)
16395: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
16396: procedure Succ(X: int64);
16397: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
16398: procedure SwapChar(var X,Y: char); //swapX follows
16399: Procedure SwapFloats( var X, Y : Float)
16400: procedure SwapGrid(grd: TStringGrid);
16401: Procedure SwapOrd( var I, J : Byte);
16402: Procedure SwapOrd( var X, Y : Integer)
16403: Procedure SwapOrd1( var I, J : Shortint);
16404: Procedure SwapOrd2( var I, J : Smallint);
16405: Procedure SwapOrd3( var I, J : Word);
16406: Procedure SwapOrd4( var I, J : Integer);
16407: Procedure SwapOrd5( var I, J : Cardinal);
16408: Procedure SwapOrd6( var I, J : Int64);
16409: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
16410: Procedure Synchronizel( Method : TMethod);
16411: procedure SyntaxCheck1Click(Sender: TObject);
16412: Procedure SysFreeString(const S: WideString); stdcall;
16413: Procedure TakeOver( Other : TLinearBitmap)
16414: Procedure Talkln(const sText: string) //async voice
16415: procedure tbtn6resClick(Sender: TObject);
16416: Procedure tbtnUseCaseClick( Sender : TObject)
16417: procedure TerminalStyle1Click(Sender: TObject);
16418: Procedure Terminate
16419: Procedure texSyntax1Click( Sender : TObject)
16420: procedure TextOut(X, Y: Integer; Text: string);
16421: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
16422: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
16423: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
16424: Procedure TextStart
16425: procedure TILE
16426: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
16427: Procedure TitleClick( Column : TColumn)
16428: Procedure ToDo
16429: procedure toolbtnTutorialClick(Sender: TObject);
16430: Procedure Trace1( AURL : string; const AResponseContent : TStream);
16431: Procedure TransferMode( ATransferMode : TIDFTPTransferMode)
16432: Procedure Truncate
16433: procedure Tutorial101Click(Sender: TObject);
16434: procedure Tutorial10Statistics1Click(Sender: TObject);
16435: procedure Tutorial11Forms1Click(Sender: TObject);
16436: procedure Tutorial12SQL1Click(Sender: TObject);
16437: Procedure tutorial1Click( Sender : TObject)
16438: Procedure tutorial21Click( Sender : TObject)
16439: Procedure tutorial31Click( Sender : TObject)
16440: Procedure tutorial4Click( Sender : TObject)
16441: Procedure Tutorial5Click( Sender : TObject)
16442: procedure Tutorial6Click(Sender: TObject);
16443: procedure Tutorial91Click(Sender: TObject);
16444: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
16445: procedure UniqueString(var str: AnsiString)
16446: procedure UnloadLoadPackage(Module: HMODULE)
16447: Procedure Unlock
16448: Procedure UNMERGE( MENU : TMAINMENU)
16449: Procedure UnRegisterChanges( Value : TChangeLink)
16450: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
16451: Procedure UnregisterConversionType( const AType : TConvType)
16452: Procedure UnRegisterProvider( Prov : TCustomProvider)
16453: Procedure UPDATE
16454: Procedure UpdateBatch( AffectRecords : TAffectRecords)
16455: Procedure UPDATECURSORPOS
16456: Procedure UpdateFile
16457: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
16458: Procedure UpdateResponse( AResponse : TWebResponse)
16459: Procedure UpdateScrollBar
16460: Procedure UpdateView1Click( Sender : TObject)
16461: procedure Val(const s: string; var n, z: Integer)
16462: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
16463: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
16464: Procedure VariantAdd( const src : Variant; var dst : Variant)
16465: Procedure VariantAnd( const src : Variant; var dst : Variant)
16466: Procedure VariantArrayRedim( var V : Variant; High : Integer)
16467: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
16468: Procedure VariantClear( var V : Variant)
16469: Procedure VariantCpy( const src : Variant; var dst : Variant)
16470: Procedure VariantDiv( const src : Variant; var dst : Variant)
16471: Procedure VariantMod( const src : Variant; var dst : Variant)
16472: Procedure VariantMul( const src : Variant; var dst : Variant)
16473: Procedure VariantOr( const src : Variant; var dst : Variant)
16474: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);

```

```

16475: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
16476: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
16477: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
16478: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
16479: Procedure VariantShl( const src : Variant; var dst : Variant)
16480: Procedure VariantShr( const src : Variant; var dst : Variant)
16481: Procedure VariantSub( const src : Variant; var dst : Variant)
16482: Procedure VariantXor( const src : Variant; var dst : Variant)
16483: Procedure VarCastError;
16484: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
16485: Procedure VarInvalidOp
16486: Procedure VarInvalidNullOp
16487: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
16488: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
16489: Procedure VarArrayCreateError
16490: Procedure VarResultCheck( AResult : HRESULT);
16491: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
16492: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
16493: Function VarTypeAsText( const AType : TVarType) : string
16494: procedure Voice(const sText: string) //async
16495: procedure Voice2(const sText: string) //sync
16496: Procedure WaitMiliSeconds( AMSec : word)
16497: Procedure WideAppend( var dst : WideString; const src : WideString)
16498: Procedure WideAssign( var dst : WideString; var src : WideString)
16499: Procedure WideDelete( var dst : WideString; index, count : Integer)
16500: Procedure WideFree( var s : WideString)
16501: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
16502: Procedure WideFromPChar( var dst : WideString; src : PChar)
16503: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
16504: Procedure WideSetLength( var dst : WideString; len : Integer)
16505: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
16506: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
16507: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
16508: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
16509: Procedure HttpGet(const Url: string; Stream:TStream);
16510: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
16511: Procedure WordWrap1Click( Sender : TObject)
16512: Procedure Write( const AOut : string)
16513: Procedure Write( Socket : TSocket)
16514: procedure Write(S: string);
16515: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
16516: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
16517: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
16518: procedure WriteBuffer(Buffer:String;Count:LongInt)
16519: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
16520: Procedure WriteChar( AValue : Char)
16521: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
16522: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
16523: Procedure WriteFloat( const Section, Name : string; Value : Double)
16524: Procedure WriteHeader( AHeader : TStrings)
16525: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
16526: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
16527: Procedure WriteLn( const AOut : string)
16528: procedure Writeln(s: string);
16529: Procedure WriteLog( const FileName, LogLine : string)
16530: Procedure WriteRFCReply( AReply : TIdRFCReply)
16531: Procedure WriteRFCStrings( AStrings : TStrings)
16532: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
16533: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
16534: Procedure WriteString( const Section, Ident, Value : string)
16535: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
16536: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
16537: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
16538: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
16539: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
16540: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
16541: procedure XMLSyntax1Click(Sender: TObject);
16542: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
16543: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
16544: Procedure ZeroFillStream( Stream : TMemoryStream)
16545: procedure XMLSyntax1Click(Sender: TObject);
16546: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
16547: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
16548: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
16549: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
16550: procedure(Sender, Source: TObject; X, Y: Integer)
16551: procedure(Sender, Target: TObject; X, Y: Integer)
16552: procedure(Sender: TObject; ASection, AWidth: Integer)
16553: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
16554: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
16555: procedure(Sender: TObject; var Action: TCloseAction)
16556: procedure(Sender: TObject; var CanClose: Boolean)
16557: procedure(Sender: TObject; var Key: Char);
16558: ProcedureName ProcedureNames ProcedureParametersCursor @
16559:
16560: *****Now Constructors constructor *****
16561: Size is: 1231 1115 996 628 550 544 501 459 (381)
16562: Attach( VersionInfoData : Pointer; Size : Integer)
16563: constructor Create( ABuckets : TBucketListSizes)

```



```

16564: Create( ACallbackWnd : HWND)
16565: Create( AClient : TCustomTaskDialog)
16566: Create( AClient : TIdTelnet)
16567: Create( ACollection : TCollection)
16568: Create( ACollection : TFavoriteLinkItems)
16569: Create( ACollection : TTaskDialogButtons)
16570: Create( AConnection : TIdCustomHTTP)
16571: Create( ACreateSuspended : Boolean)
16572: Create( ADataSet : TCustomSQLDataSet)
16573: CREATE( ADATASET : TDATASET)
16574: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
16575: Create( AGrid : TCustomDBGrid)
16576: Create( AGrid : TStringGrid; AIndex : Longint)
16577: Create( AHTTP : TIdCustomHTTP)
16578: Create( AListItems : TListItems)
16579: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
16580: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
16581: Create( AOwner : TCommonCalendar)
16582: Create( AOwner : TComponent)
16583: CREATE( AOWNER : TCOMPONENT)
16584: Create( AOwner : TCustomListView)
16585: Create( AOwner : TCustomOutline)
16586: Create( AOwner : TCustomRichEdit)
16587: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
16588: Create( AOwner : TCustomTreeView)
16589: Create( AOwner : TIdUserManager)
16590: Create( AOwner : TListItems)
16591: Create(
  AOwner:TObject;Handle:hDBICur;CBType:CBType;CBBuf:Ptr;CBBufSize:Int;CallbackEvent:TBDECallbackEvent;
  Chain:Bool)
16592: CREATE( AOWNER : TPERSISTENT)
16593: Create( AOwner : TPersistent)
16594: Create( AOwner : TTable)
16595: Create( AOwner : TTreeNode)
16596: Create( AOwner : TWinControl; const ClassName : string)
16597: Create( AParent : TIdCustomHTTP)
16598: Create( AParent : TUpdateTree; AResolver : TCustomResolver)
16599: Create( AProvider : TBaseProvider)
16600: Create( AProvider : TCustomProvider);
16601: Create( AProvider : TDataSetProvider)
16602: Create( ASocket : TCustomWinSocket; TimeOut : Longint)
16603: Create( ASocket : TSocket)
16604: Create( AStrings : TWideStrings)
16605: Create( AToolBar : TToolBar)
16606: Create( ATreeNodes : TTreeNode)
16607: Create( Autofill : boolean)
16608: Create( AWebPageInfo : TAbstractWebPageInfo)
16609: Create( AWebRequest : TWebRequest)
16610: Create( Collection : TCollection)
16611: Create( Collection : TIdMessageParts; ABody : TStrings)
16612: Create( Collection : TIdMessageParts; const AFileName : TFileName)
16613: Create( Column : TColumn)
16614: Create( const AConvFamily : TConvFamily; const ADescription : string)
16615: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
16616: Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
  AFromCommonProc : TConversionProc)
16617: Create( const AInitialState : Boolean; const AManualReset : Boolean)
16618: Create( const ATabSet : TTabSet)
16619: Create( const Compensate : Boolean)
16620: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
16621: Create( const FileName : string)
16622: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
  : Int64; const SecAttr : PSecurityAttributes);
16623: Create( const FileName : string; FileMode : WordfmShareDenyWrite)
16624: Create( const MaskValue : string)
16625: Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
16626: Create( const Prefix : string)
16627: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
16628: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
16629: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
16630: Create( CoolBar : TCoolBar)
16631: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
16632: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
16633: Create( DataSet :TDataSet; const
  Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
  DepFields : TBits; FieldMap : TFieldMap)
16634: Create( DBCtrlGrid : TDBCtrlGrid)
16635: Create( DStableProducer : TDStableProducer)
16636: Create( DStableProducer : TDStableProducer; ColumnClass : THTMLTableColumnClass)
16637: Create( ErrorCode : DBIResult)
16638: Create( Field : TBlobField; Mode : TBlobStreamMode)
16639: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
16640: Create( HeaderControl : TCustomHeaderControl)
16641: Create( HTTPRequest : TWebRequest)
16642: Create( iStart : integer; sText : string)
16643: Create( iValue : Integer)
16644: Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
16645: Create( MciErrNo : MCIERROR; const Msg : string)
16646: Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);

```

```

16647: Create( Message : string; ErrorCode : DBResult)
16648: Create( Msg : string)
16649: Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
16650: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
16651: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
16652: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
16653: Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
16654: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
16655: Create( Owner : TCustomComboBoxEx)
16656: CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
16657: Create( Owner : TPersistent)
16658: Create( Params : TStrings)
16659: Create( Size : Cardinal)
16660: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
16661: Create( StatusBar : TCustomStatusBar)
16662: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
16663: Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
16664: Create(AHandle:Integer)
16665: Create(AOwner: TComponent); virtual;
16666: Create(const AURI : string)
16667: Create(FileName:String;Mode:Word)
16668: Create(Instance:THandle;ResName:String;ResType:PChar)
16669: Create(Stream : TStream)
16670: Create( ADataset : TDataset);
16671: Create( const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
SecAttr:PSecurityAttributes);
16672: Create( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
16673: Create2( Other : TObject);
16674: CreateAt( FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
16675: CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
16676: CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
16677: CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
16678: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
16679: CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
16680: CreateRes( Ident : Integer);
16681: CreateRes( MciErrNo : MCIERROR; Ident : Integer)
16682: CreateRes( ResStringRec : PResStringRec);
16683: CreateResHelp( Ident : Integer; AHelpContext : Integer);
16684: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
16685: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
16686: CreateSize( AWidth, AHeight : Integer)
16687: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
16688:
16689: -----
16690: unit uPSI_MathMax;
16691: -----
16692: CONSTS
16693: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
16694: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
16695: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
16696: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
16697: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
16698: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(Pi)
16699: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
16700: Pi: Float = 3.1415926535897932384626433832795; // PI
16701: PI: Extended = 3.1415926535897932384626433832795;
16702: PiOn2: Float = 1.5707963267948966192313216916398; // PI / 2
16703: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
16704: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4
16705: Sqrt2: Float = 1.4142135623730950488016887242097; // Sqrt(2)
16706: Sqrt3: Float = 1.7320508075688772935274463415059; // Sqrt(3)
16707: Sqrt5: Float = 2.2360679774997896964091736687313; // Sqrt(5)
16708: Sqrt10: Float = 3.1622776601683793319988935444327; // Sqrt(10)
16709: SqrtPi: Float = 1.7724538509055160272981674833411; // Sqrt(Pi)
16710: Sqrt2Pi: Float = 2.506628274631000502415765284811; // Sqrt(2 * PI)
16711: TwoPi: Float = 6.283185307179586476925286766559; // 2 * PI
16712: ThreePi: Float = 9.4247779607693797153879301498385; // 3 * PI
16713: Ln2: Float = 0.69314718055994530941723212145818; // Ln(2)
16714: Ln10: Float = 2.3025850929940456840179914546844; // Ln(10)
16715: LnPi: Float = 1.1447298858494001741434273513531; // Ln(Pi)
16716: Log2J: Float = 0.30102999566398119521373889472449; // Log10(2)
16717: Log3: Float = 0.47712125471966243729502790325512; // Log10(3)
16718: LogPi: Float = 0.4971498726941338543512682882909; // Log10(Pi)
16719: LogE: Float = 0.43429448190325182765112891891661; // Log10(E)
16720: E: Float = 2.7182818284590452353602874713527; // Natural constant
16721: hLn2Pi: Float = 0.91893853320467274178032973640562; // Ln(2*PI)/2
16722: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
16723: TwoToPower63: Float = 9223372036854775808.0; // 2^63
16724: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
16725: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
16726: RadCor : Double = 57.29577951308232; {number of degrees in a radian}
16727: StDelta : Extended = 0.00001; {delta for difference equations}
16728: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
16729: StMaxIterations : Integer = 100; {max attempts for convergence}
16730:
16731: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
16732: begin
16733: MetersPerInch = 0.0254; // [1]
16734: MetersPerFoot = MetersPerInch * 12;

```

```

16735: MetersPerYard = MetersPerFoot * 3;
16736: MetersPerMile = MetersPerFoot * 5280;
16737: MetersPerNauticalMiles = 1852;
16738: MetersPerAstronomicalUnit = 1.49598E11; // [4]
16739: MetersPerLightSecond = 2.99792458E8; // [5]
16740: MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
16741: MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
16742: MetersPerCubit = 0.4572; // [6][7]
16743: MetersPerFathom = MetersPerFoot * 6;
16744: MetersPerFurlong = MetersPerYard * 220;
16745: MetersPerHand = MetersPerInch * 4;
16746: MetersPerPace = MetersPerInch * 30;
16747: MetersPerRod = MetersPerFoot * 16.5;
16748: MetersPerChain = MetersPerRod * 4;
16749: MetersPerLink = MetersPerChain / 100;
16750: MetersPerPoint = MetersPerInch * 0.013837; // [7]
16751: MetersPerPica = MetersPerPoint * 12;
16752:
16753: SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
16754: SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
16755: SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
16756: SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
16757: SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
16758: SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
16759:
16760: CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
16761: CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
16762: CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
16763: CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
16764: CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
16765: CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
16766: CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
16767: CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
16768:
16769: CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
16770: CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
16771: CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
16772: CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
16773: CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
16774: CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
16775: CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
16776: CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
16777:
16778: CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
16779: CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
16780: CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
16781: CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
16782: CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
16783: CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
16784:
16785: CubicMetersPerUKGallon = 0.00454609; // [2][7]
16786: CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
16787: CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
16788: CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
16789: CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
16790: CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
16791: CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
16792: CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
16793: CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
16794:
16795: GramsPerPound = 453.59237; // [1][7]
16796: GramsPerDrams = GramsPerPound / 256;
16797: GramsPerGrains = GramsPerPound / 7000;
16798: GramsPerTons = GramsPerPound * 2000;
16799: GramsPerLongTons = GramsPerPound * 2240;
16800: GramsPerOunces = GramsPerPound / 16;
16801: GramsPerStones = GramsPerPound * 14;
16802:
16803: MaxAngle', ( 922337203685477580.0);
16804: MaxTanH', ( 5678.2617031470719747459655389854);
16805: MaxFactorial', 'LongInt').SetInt( 1754);
16806: MaxFloatingPoint', (1.189731495357231765085759326628E+4932);
16807: MinFloatingPoint', (3.3621031431120935062626778173218E-4932);
16808: MaxTanH', ( 354.89135644669199842162284618659);
16809: MaxFactorial', 'LongInt').SetInt( 170);
16810: MaxFloatingPointD', (1.797693134862315907729305190789E+308);
16811: MinFloatingPointD', (2.2250738585072013830902327173324E-308);
16812: MaxTanH', ( 44.361419555836499802702855773323);
16813: MaxFactorial', 'LongInt').SetInt( 33);
16814: MaxFloatingPointS', ( 3.4028236692093846346337460743177E+38);
16815: MinFloatingPointS', ( 1.1754943508222875079687365372222E-38);
16816: PiExt', ( 3.1415926535897932384626433832795);
16817: RatioDegToRad', ( PiExt / 180.0);
16818: RatioGradToRad', ( PiExt / 200.0);
16819: RatioDegToGrad', ( 200.0 / 180.0);
16820: RatioGradToDeg', ( 180.0 / 200.0);
16821: Crc16PolynomCCITT', 'LongWord').SetUInt( $1021);
16822: Crc16PolynomIBM', 'LongWord').SetUInt( $8005);
16823: Crc16Bits', 'LongInt').SetInt( 16);

```

```

16824: Crc16Bytes', 'LongInt').SetInt( 2);
16825: Crc16HighBit', 'LongWord').SetUInt( $8000);
16826: NotCrc16HighBit', 'LongWord').SetUInt( $7FFF);
16827: Crc32PolynomIEEE', 'LongWord').SetUInt( $04C11DB7);
16828: Crc32PolynomCastagnoli', 'LongWord').SetUInt( $1EDC6F41);
16829: Crc32Koopman', 'LongWord').SetUInt( $741B8CD7);
16830: Crc32Bits', 'LongInt').SetInt( 32);
16831: Crc32Bytes', 'LongInt').SetInt( 4);
16832: Crc32HighBit', 'LongWord').SetUInt( $80000000);
16833: NotCrc32HighBit', 'LongWord').SetUInt( $7FFFFFFF);
16834:
16835: MinByte      = Low(Byte);
16836: MaxByte      = High(Byte);
16837: MinWord      = Low(Word);
16838: MaxWord      = High(Word);
16839: MinShortInt  = Low(ShortInt);
16840: MaxShortInt  = High(ShortInt);
16841: MinSmallInt  = Low(SmallInt);
16842: MaxSmallInt  = High(SmallInt);
16843: MinLongWord  = LongWord(Low(LongWord));
16844: MaxLongWord  = LongWord(High(LongWord));
16845: MinLongInt   = LongInt(Low(LongInt));
16846: MaxLongInt   = LongInt(High(LongInt));
16847: MinInt64     = Int64(Low(Int64));
16848: MaxInt64     = Int64(High(Int64));
16849: MinInteger   = Integer(Low(Integer));
16850: MaxInteger   = Integer(High(Integer));
16851: MinCardinal  = Cardinal(Low(Cardinal));
16852: MaxCardinal  = Cardinal(High(Cardinal));
16853: MinNativeUInt = NativeUInt(Low(NativeUInt));
16854: MaxNativeUInt = NativeUInt(High(NativeUInt));
16855: MinNativeInt  = NativeInt(Low(NativeInt));
16856: MaxNativeInt  = NativeInt(High(NativeInt));
16857: Function CosH( const Z : Float) : Float;
16858: Function SinH( const Z : Float) : Float;
16859: Function TanH( const Z : Float) : Float;
16860:
16861:
16862: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
16863: InvLn2      = 1.44269504088896340736; { 1/Ln(2) }
16864: InvLn10     = 0.43429448190325182765; { 1/Ln(10) }
16865: TwoPi       = 6.28318530717958647693; { 2*Pi }
16866: PiDiv2      = 1.57079632679489661923; { Pi/2 }
16867: SqrtPi      = 1.77245385090551602730; { Sqrt(Pi) }
16868: Sqrt2Pi     = 2.50662827463100050242; { Sqrt(2*Pi) }
16869: InvSqrt2Pi  = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
16870: LnSqrt2Pi   = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
16871: Ln2PiDiv2   = 0.91893853320467274178; { Ln(2*Pi)/2 }
16872: Sqrt2       = 1.41421356237309504880; { Sqrt(2) }
16873: Sqrt2Div2   = 0.70710678118654752440; { Sqrt(2)/2 }
16874: Gold        = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
16875: CGold       = 0.38196601125010515179; { 2 - GOLD }
16876: MachEp      = 2.220446049250313E-16; { 2^(-52) }
16877: MaxNum      = 1.797693134862315E+308; { 2^1024 }
16878: MinNum      = 2.225073858507202E-308; { 2^(-1022) }
16879: MaxLog      = 709.7827128933840;
16880: MinLog      = -708.3964185322641;
16881: MaxFac      = 170;
16882: MaxGam      = 171.624376956302;
16883: MaxLgm      = 2.556348E+305;
16884: SingleCompareDelta = 1.0E-34;
16885: DoubleCompareDelta = 1.0E-280;
16886: { $IFDEF CLR }
16887: ExtendedCompareDelta = DoubleCompareDelta;
16888: { $ELSE }
16889: ExtendedCompareDelta = 1.0E-4400;
16890: { $ENDIF }
16891: Bytes1KB    = 1024;
16892: Bytes1MB    = 1024 * Bytes1KB;
16893: Bytes1GB    = 1024 * Bytes1MB;
16894: Bytes64KB   = 64 * Bytes1KB;
16895: Bytes64MB   = 64 * Bytes1MB;
16896: Bytes2GB    = 2 * LongWord(Bytes1GB);
16897: clBlack32', $FF000000 );
16898: clDimGray32', $FF3F3F3F );
16899: clGray32', $FF7F7F7F );
16900: clLightGray32', $FFBFBFBF );
16901: clWhite32', $FFFFFFF );
16902: clMaroon32', $FF7F0000 );
16903: clGreen32', $FF007F00 );
16904: clOlive32', $FF7F7F00 );
16905: clNavy32', $FF00007F );
16906: clPurple32', $FF7F007F );
16907: clTeal32', $FF007F7F );
16908: clRed32', $FFFF0000 );
16909: clLime32', $FF00FF00 );
16910: clYellow32', $FFFFFF00 );
16911: clBlue32', $FF0000FF );
16912: clFuchsia32', $FFFF00FF );

```



```
16913: clAqua32', $FFF0FFFF );
16914: clAliceBlue32', $FFF0F8FF );
16915: clAntiqueWhite32', $FFFAEBD7 );
16916: clAquamarine32', $FFF7FDD4 );
16917: clAzure32', $FFF0FFFF );
16918: clBeige32', $FFF5F5DC );
16919: clBisque32', $FFFFE4C4 );
16920: clBlancheDalmond32', $FFFEBECD );
16921: clBlueViolet32', $FF8A2BE2 );
16922: clBrown32', $FFA52A2A );
16923: clBurlyWood32', $FFDEB887 );
16924: clCadetblue32', $FF5F9EA0 );
16925: clChartReuse32', $FF7FFF00 );
16926: clChocolate32', $FFD2691E );
16927: clCoral32', $FFFF7F50 );
16928: clCornFlowerBlue32', $FF6495ED );
16929: clCornSilk32', $FFFFFF8DC );
16930: clCrimson32', $FFDC143C );
16931: clDarkBlue32', $FF00008B );
16932: clDarkCyan32', $FF008B8B );
16933: clDarkGoldenRod32', $FFB8860B );
16934: clDarkGray32', $FFA9A9A9 );
16935: clDarkGreen32', $FF006400 );
16936: clDarkGrey32', $FFA9A9A9 );
16937: clDarkKhaki32', $FFBDB76B );
16938: clDarkMagenta32', $FF8B008B );
16939: clDarkOliveGreen32', $FF556B2F );
16940: clDarkOrange32', $FFFF8C00 );
16941: clDarkOrchid32', $FF9932CC );
16942: clDarkRed32', $FF8B0000 );
16943: clDarkSalmon32', $FFE9967A );
16944: clDarkSeaGreen32', $FF8FBC8F );
16945: clDarkSlateBlue32', $FF483D8B );
16946: clDarkSlateGray32', $FF2F4F4F );
16947: clDarkSlateGrey32', $FF2F4F4F );
16948: clDarkTurquoise32', $FF00CED1 );
16949: clDarkViolet32', $FF9400D3 );
16950: clDeepPink32', $FFFF1493 );
16951: clDeepSkyBlue32', $FF00BFFF );
16952: clDodgerBlue32', $FF1E90FF );
16953: clFireBrick32', $FFB22222 );
16954: clFloralWhite32', $FFFFFFAF0 );
16955: clGainsboro32', $FFDCDCDC );
16956: clGhostWhite32', $FFF8F8FF );
16957: clGold32', $FFFFD700 );
16958: clGoldenRod32', $FFDAA520 );
16959: clGreenYellow32', $FFADFF2F );
16960: clGrey32', $FF808080 );
16961: clHoneyDew32', $FFF0FFFF );
16962: clHotPink32', $FFF669B4 );
16963: clIndianRed32', $FFCD5C5C );
16964: clIndigo32', $FF4B0082 );
16965: clIvory32', $FFFFFFF0 );
16966: clKhaki32', $FFF0E68C );
16967: clLavender32', $FFE6E6FA );
16968: clLavenderBlush32', $FFF0F0F5 );
16969: clLawnGreen32', $FF7CFC00 );
16970: clLemonChiffon32', $FFFFFFACD );
16971: clLightBlue32', $FFADD8E6 );
16972: clLightCoral32', $FFF08080 );
16973: clLightCyan32', $FFE0FFFF );
16974: clLightGoldenRodYellow32', $FFFAFAD2 );
16975: clLightGreen32', $FF90EE90 );
16976: clLightGrey32', $FFD3D3D3 );
16977: clLightPink32', $FFFB6C1 );
16978: clLightSalmon32', $FFFA07A );
16979: clLightSeagreen32', $FF20B2AA );
16980: clLightSkyblue32', $FF87CEFA );
16981: clLightSlategray32', $FF778899 );
16982: clLightSlategrey32', $FF778899 );
16983: clLightSteelblue32', $FFB0C4DE );
16984: clLightYellow32', $FFFFFFF0 );
16985: clLtGray32', $FFC0C0C0 );
16986: clMedGray32', $FFA0A0A4 );
16987: clDkGray32', $FF808080 );
16988: clMoneyGreen32', $FFC0DCC0 );
16989: clLegacySkyBlue32', $FFA6CAF0 );
16990: clCream32', $FFFFFFBF0 );
16991: clLimeGreen32', $FF32CD32 );
16992: clLinen32', $FFFAF0E6 );
16993: clMediumAquamarine32', $FF66CDAA );
16994: clMediumBlue32', $FF0000CD );
16995: clMediumOrchid32', $FFBA55D3 );
16996: clMediumPurple32', $FF9370DB );
16997: clMediumSeaGreen32', $FF3CB371 );
16998: clMediumSlateBlue32', $FF7B68EE );
16999: clMediumSpringGreen32', $FF00FA9A );
17000: clMediumTurquoise32', $FF48D1CC );
17001: clMediumVioletRed32', $FFC71585 );
```

```

17002:   clMidnightBlue32', $FF191970 ));
17003:   clMintCream32', $FFF5FFFA ));
17004:   clMistyRose32', $FFFFE4E1 ));
17005:   clMoccasin32', $FFFFE4B5 ));
17006:   clNavajoWhite32', $FFFFDEAD ));
17007:   clOldLace32', $FFFDF5E6 ));
17008:   clOliveDrab32', $FF6B8E23 ));
17009:   clOrange32', $FFFA500 ));
17010:   clOrangeRed32', $FFFF4500 ));
17011:   clOrchid32', $FFDA70D6 ));
17012:   clPaleGoldenRod32', $FFEE8AA ));
17013:   clPaleGreen32', $FF98FB98 ));
17014:   clPaleTurquoise32', $FFAFEEEE ));
17015:   clPaleVioletred32', $FFDB7093 ));
17016:   clPapayaWhip32', $FFFFFED5 ));
17017:   clPeachPuff32', $FFFDAB9 ));
17018:   clPeru32', $FFCD853F ));
17019:   clPlum32', $FFDDA0DD ));
17020:   clPowderBlue32', $FFB0E0E6 ));
17021:   clRosyBrown32', $FFBC8F8F ));
17022:   clRoyalBlue32', $FF4169E1 ));
17023:   clSaddleBrown32', $FF8B4513 ));
17024:   clSalmon32', $FFFA8072 ));
17025:   clSandyBrown32', $FFFA4A60 ));
17026:   clSeaGreen32', $FF2E8B57 ));
17027:   clSeaShell32', $FFFFFFEE ));
17028:   clSienna32', $FFA0522D ));
17029:   clSilver32', $FFC0C0C0 ));
17030:   clSkyblue32', $FF87CEEB ));
17031:   clSlateBlue32', $FF6A5ACD ));
17032:   clSlateGray32', $FF708090 ));
17033:   clSlateGrey32', $FF708090 ));
17034:   clSnow32', $FFFFFFFA ));
17035:   clSpringgreen32', $FF00FF7F ));
17036:   clSteelblue32', $FF4682B4 ));
17037:   clTan32', $FFD2B48C ));
17038:   clThistle32', $FFD8BFD8 ));
17039:   clTomato32', $FFFF6347 ));
17040:   clTurquoise32', $FF40E0D0 ));
17041:   clViolet32', $FFEE82EE ));
17042:   clWheat32', $FFF5DEB3 ));
17043:   clWhitesmoke32', $FFF5F5F5 ));
17044:   clYellowgreen32', $FF9ACD32 ));
17045:   clTrWhite32', $7FFFFFFF ));
17046:   clTrBlack32', $7F000000 ));
17047:   clTrRed32', $7FFF0000 ));
17048:   clTrGreen32', $7F00FF00 ));
17049:   clTrBlue32', $7F0000FF ));
17050:   // Fixed point math constants
17051:   FixedOne = $10000; FixedHalf = $7FFF;
17052:   FixedPI = Round(PI * FixedOne);
17053:   FixedToFloat = 1/FixedOne;
17054:
17055:   Special Types
17056:   *****
17057:   type Complex = record
17058:     X, Y : Float;
17059:   end;
17060:   type TVector = array of Float;
17061:   TIntVector = array of Integer;
17062:   TCompVector = array of Complex;
17063:   TBoolVector = array of Boolean;
17064:   TStrVector = array of String;
17065:   TMatrix = array of TVector;
17066:   TIntMatrix = array of TIntVector;
17067:   TCompMatrix = array of TCompVector;
17068:   TBoolMatrix = array of TBoolVector;
17069:   TStrMatrix = array of TStrVector;
17070:   TByteArray = array[0..32767] of byte; !
17071:   THexArray = array[0..15] of Char; // = '0123456789ABCDEF';
17072:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
17073:   T2StringArray = array of array of string;
17074:   T2IntegerArray = array of array of integer;
17075:   AddTypeS('INT_PTR', 'Integer
17076:   AddTypeS('LONG_PTR', 'Integer
17077:   AddTypeS('UINT_PTR', 'Cardinal
17078:   AddTypeS('ULONG_PTR', 'Cardinal
17079:   AddTypeS('DWORD_PTR', 'ULONG_PTR
17080:   TIntegerDynArray', 'array of Integer
17081:   TCardinalDynArray', 'array of Cardinal
17082:   TWordDynArray', 'array of Word
17083:   TSmallIntDynArray', 'array of SmallInt
17084:   TByteDynArray', 'array of Byte
17085:   TShortIntDynArray', 'array of ShortInt
17086:   TInt64DynArray', 'array of Int64
17087:   TLongWordDynArray', 'array of LongWord
17088:   TSingleDynArray', 'array of Single
17089:   TDoubleDynArray', 'array of Double
17090:   TBooleanDynArray', 'array of Boolean

```

```

17091: TStringDynArray', 'array of string
17092: TWideStringDynArray', 'array of WideString
17093: TDynByteArray = array of Byte;
17094: TDynShortintArray = array of Shortint;
17095: TDynSmallintArray = array of Smallint;
17096: TDynWordArray = array of Word;
17097: TDynIntegerArray = array of Integer;
17098: TDynLongintArray = array of Longint;
17099: TDynCardinalArray = array of Cardinal;
17100: TDynInt64Array = array of Int64;
17101: TDynExtendedArray = array of Extended;
17102: TDynDoubleArray = array of Double;
17103: TDynSingleArray = array of Single;
17104: TDynFloatArray = array of Float;
17105: TDynPointerArray = array of Pointer;
17106: TDynStringArray = array of string;
17107: TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
17108:   ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
17109: TSynSearchOptions = set of TSynSearchOption;
17110:
17111:
17112:
17113: /** Project : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
17114: -----
17115: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
17116: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
17117: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
17118: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
17119: function CheckStringSum(vstring: string): integer;
17120: function HexToInt(HexNum: string): LongInt;
17121: function IntToBin(Int: Integer): String;
17122: function BinToInt(Binary: String): Integer;
17123: function HexToBin(HexNum: string): string; external2
17124: function BinToHex(Binary: String): string;
17125: function IntToFloat(i: Integer): double;
17126: function AddThousandSeparator(S: string; myChr: Char): string;
17127: function Max3(const X,Y,Z: Integer): Integer;
17128: procedure Swap(var X,Y: char); // faster without inline
17129: procedure ReverseString(var S: String);
17130: function CharToHexStr(Value: Char): string;
17131: function CharToUnicode(Value: Char): string;
17132: function Hex2Dec(Value: Str002): Byte;
17133: function HexStrCodeToStr(Value: string): string;
17134: function HexToStr(i: integer; value: string): string;
17135: function UnicodeToStr(Value: string): string;
17136: function CRC16(statement: string): string;
17137: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
17138: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
17139: procedure ShowInterfaces(myFile: string);
17140: function Fact2(av: integer): extended;
17141: function BoolToStr(B: Boolean): string;
17142: function GCD(x, y : LongInt) : LongInt;
17143: function LCM(m,n: longint): longint;
17144: function GetASCII: string;
17145: function GetItemHeight(Font: TFont): Integer;
17146: function myPlaySound(s: pchar; flag, syncflag: integer): boolean;
17147: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
17148: function getHINSTANCE: longword;
17149: function getHMODULE: longword;
17150: function GetASCII: string;
17151: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
17152: function WordIsOk(const AWord: string; var VW: Word): boolean;
17153: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
17154: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
17155: function SafeStr(const s: string): string;
17156: function ExtractUrlPath(const FileName: string): string;
17157: function ExtractUrlName(const FileName: string): string;
17158: function IsInternet: boolean;
17159: function RotateLeft1Bit_u32( Value: uint32): uint32;
17160: procedure LinearRegression(const KnownY: array of Double; const KnownX: array of Double; NData: Int; var
  LF: TStLinEst; ErrorStats : Boolean);
17161: procedure getEnvironmentInfo;
17162: procedure AntiFreeze;
17163: function GetCPUSpeed: Double;
17164: function IsVirtualPcGuest : Boolean;
17165: function IsVmWareGuest : Boolean;
17166: procedure StartSerialDialog;
17167: function IsWoW64: boolean;
17168: function IsWow64String(var s: string): Boolean;
17169: procedure StartThreadDemo;
17170: function RGB(R,G,B: Byte): TColor;
17171: function SendLn(amess: string): boolean;
17172: procedure maXbox;
17173: function AspectRatio(aWidth, aHeight: Integer): String;
17174: function wget(aURL, afile: string): boolean;
17175: procedure PrintList(Value: TStringList);
17176: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
17177: procedure getEnvironmentInfo;
17178: procedure AntiFreeze;

```

```

17179: function getBitmap(apat: string): TBitmap;
17180: procedure ShowMessageBig(const aText : string);
17181: function YesNoDialog(const ACaption, AMsg: string): boolean;
17182: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
17183: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
17184: //function myStrToBytes(const Value: String): TBytes;
17185: //function myBytesToStr(const Value: TBytes): String;
17186: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
17187: function getBitmap(apat: string): TBitmap;
17188: procedure ShowMessageBig(const aText : string);
17189: function StrToBytes(const Value: String): TBytes;
17190: function BytesToStr(const Value: TBytes): String;
17191: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
17192: function ReversedDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var
    HostName:String):Bool;
17193: function FindInPaths(const fileName, paths : String) : String;
17194: procedure initHexArray(var hexn: THexArray);
17195: function josephusG(n,k: integer; var graphout: string): integer;
17196: function isPowerof2(num: int64): boolean;
17197: function powerOf2(exponent: integer): int64;
17198: function getBigPI: string;
17199: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
17200: function GetASCIILine: string;
17201: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
    pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
17202:
17203: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
17204: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
17205: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
17206: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
17207: function isKeyPressed: boolean;
17208: function Keypress: boolean;
17209: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
17210: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
17211: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
17212: function GetOSName: string;
17213: function GetOSVersion: string;
17214: function GetOSNumber: string;
17215: function getEnvironmentString: string;
17216: procedure StrReplace(var Str: String; Old, New: String);
17217: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
17218: function getTeamViewerID: string;
17219: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
17220: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
17221: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
17222: procedure GetQRCode2(Width,Height: Word;Correct_Level:string;const Data:string;apat:string);
17223: function StartSocketService: Boolean;
17224: procedure StartSocketServiceForm;
17225: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
17226: function GetFileList1(apat: string): TStringlist;
17227: procedure LetFileList(FileList: TStringlist; apath: string);
17228: procedure StartWeb(aurl: string);
17229: function GetTodayFiles(startdir, amask: string): TStringlist;
17230: function PortTCPisOpen(dwPort : Word; ipAddressStr: String): boolean;
17231: function JavahashCode(val: string): Integer;
17232: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
17233: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
17234: Procedure HideWindowForSeconds(secs: integer); { //3 seconds}
17235: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); { //3 seconds}
17236: Procedure ConvertToGray(Cnv: TCanvas);
17237: function GetFileDate(aFile:string; aWithTime:Boolean):string;
17238: procedure ShowMemory;
17239: function ShowMemory2: string;
17240: function getHostIP: string;
17241: procedure ShowBitmap(bmap: TBitmap);
17242: function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
17243:
17244:
17245: // News of 3.9.8 up
17246: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
17247: Conversion Routines, Prebuild Forms, more RCDData, DebugOutString
17248: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
17249: JvChart - TjvChart Component - 2009 Public
17250: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
17251: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
17252: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
17253: DMATH DLL included incl. Demos
17254: Interface Navigator menu/View/Intf Navigator
17255: Unit Explorer menu/Debug/Units Explorer
17256: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maxXcel
17257: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
17258: Script History to 9 Files WebServer light /Options/Addons/WebServer
17259: Full Text Finder, JVSIMLogic Simulator Package
17260: Halt-Stop Program in Menu, WebServer2, Stop Event ,
17261: Conversion Routines, Prebuild Forms, CodeSearch
17262: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
17263: Conversion Routines, Prebuild Forms, more RCDData, DebugOutString
17264: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
17265: JvChart - TjvChart Component - 2009 Public, JvgXMLSerializer, TjvPaintFX
17266: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib

```



```

17267: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
17268: IDE Reflection API, Session Service Shell S3
17269: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
17270: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
17271: arduino map() function, PMRandom Generator
17272: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
17273: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
17274: REST Test Lib, Multilang Component, Forth Interpreter
17275: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
17276: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
17277: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
17278: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
17279: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
17280: QRCode Service, add more CFunctions like CDateTime of Synapse
17281: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
17282: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
17283: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
17284: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
17285: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
17286: BOLD Package, Indy Package5, maTRIX. MATHEMAX
17287: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
17288: emacs layers: system-package-component-unit-class-function-block
17289: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
17290: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
17291: Tutorial 18.3 RGB LED, OpenGL Geometry, maxpix, statictext
17292: OpenGL Game Demo: ..Options/Add Ons/Reversi
17293: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
17294: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
17295: 7% performance gain (hot spot profiling)
17296: PEP -Pascal Education Program, GSM Module, CGI, PHP Runner
17297: add 42 + 22 (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
17298: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
17299: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
17300:
17301: add routines in 3.9.7.5
17302: 097: procedure RIRegister_BarCodeScanner_Routines(S: TPSExec);
17303: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
17304: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
17305: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
17306: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
17307: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
17308: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
17309:
17310: ////////////////////////////////// TestUnits //////////////////////////////////
17311: SelftestPEM;
17312: SelfTestCFundamentUtils;
17313: SelfTestCFileUtils;
17314: SelfTestCDateTime;
17315: SelfTestCTimer;
17316: SelfTestCRandom;
17317: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
17318: Assert(WinPathToUnixPath('c\d.f') = '/c/d.f', 'WinPathToUnixPath
17319:
17320: // Note: There's no need for installing a client certificate in the
17321: // webbrowser. The server asks the webbrowser to send a certificate but
17322: // if nothing is installed the software will work because the server
17323: // doesn't check to see if a client certificate was supplied. If you want you can install:
17324: //
17325: // file: c_cacert.pl2
17326: // password: c_cakey
17327:
17328: TGraphicControl = class(TControl)
17329: private
17330: FCanvas: TCanvas;
17331: procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
17332: protected
17333: procedure Paint; virtual;
17334: property Canvas: TCanvas read FCanvas;
17335: public
17336: constructor Create(AOwner: TComponent); override;
17337: destructor Destroy; override;
17338: end;
17339:
17340: TCustomControl = class(TWinControl)
17341: private
17342: FCanvas: TCanvas;
17343: procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
17344: protected
17345: procedure Paint; virtual;
17346: procedure PaintWindow(DC: HDC); override;
17347: property Canvas: TCanvas read FCanvas;
17348: public
17349: constructor Create(AOwner: TComponent); override;
17350: destructor Destroy; override;
17351: end;
17352: RegisterPublishedProperties;
17353: ('ONCHANGE', 'TNotifyEvent', iptrw);
17354: ('ONCLICK', 'TNotifyEvent', iptrw);
17355: ('ONDBLCLICK', 'TNotifyEvent', iptrw);

```

```

17356: ('ONENTER', 'TNotifyEvent', iptrw);
17357: ('ONEXIT', 'TNotifyEvent', iptrw);
17358: ('ONKEYDOWN', 'TKeyEvent', iptrw);
17359: ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
17360: ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
17361: ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
17362: ('ONMOUSEUP', 'TMouseEvent', iptrw);
17363: //*****
17364: // To stop the while loop, click on Options/Show Include (boolean switch)!
17365: Control a loop in a script with a form event:
17366: IncludeON; //control the while loop
17367: while maxform1.ShowInclude1.checked do begin //menu event Options/Show Include
17368:
17369: //-----
17370: //*****mX4 ini-file Configuration*****
17371: //-----
17372: using config file maxboxdef.ini menu/Help/Config File
17373:
17374: /*** Definitions for maXbox mX3 ***/
17375: [FORM]
17376: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
17377: FONTSIZE=14
17378: EXTENSION=txt
17379: SCREENX=1386
17380: SCREENY=1077
17381: MEMHEIGHT=350
17382: PRINTFONT=Courier New //GUI Settings
17383: LINENUMBERS=Y //line numbers at gutter in editor at left side
17384: EXCEPTIONLOG=Y //store excepts and success in 2 log files see below! - menu Debug/Show Last Exceptions
17385: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
17386: BOOTSCRIPT=Y //enabling load a boot script
17387: MEMORYREPORT=Y //shows memory report on closing maXbox
17388: MACRO=Y //expand macros (see below) in code e.g. #path:E:\maxbox\maxbox3\docs\
17389: NAVIGATOR=N //shows function list at the right side of editor
17390: NAVWIDTH=350 //width of the right side interface list <CTRL L>
17391: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
17392: [WEB]
17393: IPPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
17394: IPHOST=192.168.1.53
17395: ROOTCERT=filepathY
17396: SCERT=filepathY
17397: RSAKEY=filepathY
17398: VERSIONCHECK=Y
17399:
17400: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
17401:
17402: Also possible to set report memory in script to override ini setting
17403: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
17404:
17405:
17406: After Change the ini file you can reload the file with ../Help/Config Update
17407:
17408: //-----
17409: //*****mX4 maildef.ini ini-file Configuration*****
17410: //-----
17411: /*** Definitions for maXMail ***/
17412: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
17413: [MAXMAIL]
17414: HOST=getmail.softwareschule.ch
17415: USER=mailusername
17416: PASS=password
17417: PORT=110
17418: SSL=Y
17419: BODY=Y
17420: LAST=5
17421:
17422: ADO Connection String:
17423: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
17424:
17425: OpenSSL Lib: unit ssl_openssl_lib;
17426: {$IFDEF CIL}
17427: const
17428: {$IFDEF LINUX}
17429: DLLSSLName = 'libssl.so';
17430: DLLUtilName = 'libcrypto.so';
17431: {$ELSE}
17432: DLLSSLName = 'ssleay32.dll';
17433: DLLUtilName = 'libeay32.dll';
17434: {$ENDIF}
17435: {$ELSE}
17436: var
17437: {$IFDEF MSWINDOWS}
17438: {$IFDEF DARWIN}
17439: DLLSSLName: string = 'libssl.dylib';
17440: DLLUtilName: string = 'libcrypto.dylib';
17441: {$ELSE}
17442: DLLSSLName: string = 'libssl.so';
17443: DLLUtilName: string = 'libcrypto.so';
17444: {$ENDIF}

```

```

17445: { $ELSE }
17446: DLLSSLName: string = 'ssleay32.dll';
17447: DLLSSLName2: string = 'libssl32.dll';
17448: DLLUtilName: string = 'libeay32.dll';
17449: { $ENDIF }
17450: { $ENDIF }
17451:
17452:
17453:
17454: //-----
17455: //*****mX4 Macro Tags *****
17456: //-----
17457:
17458: asm #name #hostmAPSN2APSN2111e, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
17459:
17460: //Tag Macros
17461:
17462: asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
17463:
17464: //Tag Macros
17465: 10188: SearchAndCopy(memol.lines, '#name', getUsernameWin, 11);
17466: 10189: SearchAndCopy(memol.lines, '#date', datetimetToStr(now), 11);
17467: 10190: SearchAndCopy(memol.lines, '#host', getComputerNameWin, 11);
17468: 10191: SearchAndCopy(memol.lines, '#path', fpath, 11);
17469: 10192: SearchAndCopy(memol.lines, '#file', fname, 11);
17470: 10199: SearchAndCopy(memol.lines, '#fils', fname + ' +SHA1(Act_Filename), 11);
17471: 10193: SearchAndCopy(memol.lines, '#locs', intToStr(getCodeEnd), 11);
17472: 10194: SearchAndCopy(memol.lines, '#perf', perftime, 11);
17473: 10195: SearchAndCopy(memol.lines, '#sign', Format('%s: %s: %s',
17474: [getUserNameWin, getComputerNameWin, datetimetToStr(now)], 11);
17475: 10196: SearchAndCopy(memol.lines, '#head', Format('%s: %s: %s %s ',
17476: [getUserNameWin, getComputerNameWin, datetimetToStr(now), Act_Filename]), 11);
17477: [getUserNameWin, getComputerNameWin, datetimetToStr(now), Act_Filename]), 11);
17478: 10198: SearchAndCopy(memol.lines, '#tech', Format('perf: %s threads: %d %s %s',
17479: [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
17480: 10298: SearchAndCopy(memol.lines, '#net', Format('DNS: %s; local IPs: %s; local IP: %s',
17481: [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
17482:
17483: // #tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
17484:
17485: //Replace Macros
17486: SearchAndCopy(memol.lines, '<TIME>', timetoStr(time), 6);
17487: SearchAndCopy(memol.lines, '<DATE>', datetetoStr(date), 6);
17488: SearchAndCopy(memol.lines, '<PATH>', fpath, 6);
17489: SearchAndCopy(memol.lines, '<EXEPATH>', EXEPath, 9);
17490: SearchAndCopy(memol.lines, '<FILE>', fname, 6);
17491: SearchAndCopy(memol.lines, '<SOURCE>', ExePath+'Source', 8);
17492:
17493: 10198: SearchAndCopy(memol.lines, '#tech!perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
17494: [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
17495: // #tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
17496: SearchAndCopy(memol.lines, 'maxbox_extract_funclist399.txt
17497:
17498: //-----
17499: //*****mX4 ToDo List Tags ../Help/ToDo List*****
17500: //-----
17501:
17502: while I < sl.Count do begin
17503: // if MatchesMask(sl[I], '/*? TODO ([a-z0-9_]*#[1-9#]*:*)') then
17504: if MatchesMask(sl[I], '/*? TODO (?*#?#)*:*)') then
17505: BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
17506: else if MatchesMask(sl[I], '/*? DONE (?*#?#)*:*)') then
17507: BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
17508: else if MatchesMask(sl[I], '/*? TODO (#?#)*:*)') then
17509: BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
17510: else if MatchesMask(sl[I], '/*? DONE (#?#)*:*)') then
17511: BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
17512: else if MatchesMask(sl[I], '/*?*TODO*:*)') then
17513: BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
17514: else if MatchesMask(sl[I], '/*?*DONE*:*)') then
17515: BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
17516: Inc(I);
17517: end;
17518:
17519:
17520: //-----
17521: //*****mX4 Public Tools API *****
17522: //-----
17523: file : unit uPSI_fMain.pas; [ $OTAP ] Open Tools API Catalog
17524: // Those functions concern the editor and preprocessor, all of the IDE
17525: Example: Call it with maxform1.Info1Click(self)
17526: Note: Call all Methods with maxForm1., e.g.:
17527: maxForm1.ShellStyle1Click(self);
17528:
17529: procedure SIRegister_fMain(CL: TPSPascalCompiler);
17530: begin
17531: Const('BYTECODE', 'String').SetString( 'bytecode.txt
17532: Const('PSTEXT', 'String').SetString( 'PS Scriptfiles (*.txt)|*.TXT

```

```

17533: Const('PSMODEL','String').SetString('PS Modelfiles (*.uc)|*.UC
17534: Const('PSPASCAL','String').SetString('PS Pascalfiles (*.pas)|*.PAS
17535: Const('PSINC','String').SetString('PS Includes (*.inc)|*.INC
17536: Const('DEFFILENAME','String').SetString('firstdemo.txt
17537: Const('DEFINIFILE','String').SetString('maxboxdef.ini
17538: Const('EXCEPTLOGFILE','String').SetString('maxboxerrorlog.txt
17539: Const('ALLFUNCTIONSLIST','String').SetString('upsi_allfunctionslist.txt
17540: Const('ALLFUNCTIONSLISTPDF','String').SetString('maxbox_functions_all.pdf
17541: Const('ALLOBJECTSLIST','String').SetString('docs\VCL.pdf
17542: Const('ALLRESOURCELIST','String').SetString('docs\upsi_allresourcelist.txt
17543: Const('ALLUNITLIST','String').SetString('docs\maxbox3_9.xml');
17544: Const('INCLUDEBOX','String').SetString('pas_includebox.inc
17545: Const('BOOTSCRIPT','String').SetString('maxbootscript.txt
17546: Const('MBVERSION','String').SetString('3.9.9.92
17547: Const('VERSION','String').SetString('3.9.9.92
17548: Const('MBVER','String').SetString('399
17549: Const('MBVERI','Integer').SetInt(399);
17550: Const('MBVERIALI','Integer').SetInt(39992);
17551: Const('EXENAME','String').SetString('maxbox3.exe
17552: Const('MXSITE','String').SetString('http://www.softwareschule.ch/maxbox.htm
17553: Const('MXVERSIONFILE','String').SetString('http://www.softwareschule.ch/maxvfile.txt
17554: Const('MXINTERNETCHECK','String').SetString('www.ask.com
17555: Const('MXMAIL','String').SetString('max@kleiner.com
17556: Const('TAB','Char').SetString('#$09);
17557: Const('CODECOMPLETION','String').SetString('bds_delphi.dci
17558: SIRegister_TMaxForm1(CL);
17559: end;
17560:
17561: with FindClass('TForm'),'TMaxForm1' do begin
17562:     memo2, 'TMemo', iptrw);
17563:     memol, 'TSynMemo', iptrw);
17564:     CB1SCList, 'TComboBox', iptrw);
17565:     mxNavigator, 'TComboBox', iptrw);
17566:     IPhost, 'string', iptrw);
17567:     IPPort, 'integer', iptrw);
17568:     COMPort, 'integer', iptrw); //3.9.6.4
17569:     Splitter1, 'TSplitter', iptrw);
17570:     PSScript, 'TPSScript', iptrw);
17571:     PS3DllPlugin, 'TPSDllPlugin', iptrw);
17572:     MainMenu1, 'TMainMenu', iptrw);
17573:     Program1, 'TMenuItem', iptrw);
17574:     Compile1, 'TMenuItem', iptrw);
17575:     Files1, 'TMenuItem', iptrw);
17576:     open1, 'TMenuItem', iptrw);
17577:     Save2, 'TMenuItem', iptrw);
17578:     Options1, 'TMenuItem', iptrw);
17579:     Savebefore1, 'TMenuItem', iptrw);
17580:     Largefont1, 'TMenuItem', iptrw);
17581:     sBytecode1, 'TMenuItem', iptrw);
17582:     Saveas3, 'TMenuItem', iptrw);
17583:     Clear1, 'TMenuItem', iptrw);
17584:     Slinenumbers1, 'TMenuItem', iptrw);
17585:     About1, 'TMenuItem', iptrw);
17586:     Search1, 'TMenuItem', iptrw);
17587:     SynPasSyn1, 'TSynPasSyn', iptrw);
17588:     memol, 'TSynMemo', iptrw);
17589:     SynEditSearch1, 'TSynEditSearch', iptrw);
17590:     WordWrap1, 'TMenuItem', iptrw);
17591:     XPManifest1, 'TXPManifest', iptrw);
17592:     SearchNext1, 'TMenuItem', iptrw);
17593:     Replace1, 'TMenuItem', iptrw);
17594:     PSImport_Controls1, 'TPSImport_Controls', iptrw);
17595:     PSImport_Classes1, 'TPSImport_Classes', iptrw);
17596:     ShowInclude1, 'TMenuItem', iptrw);
17597:     SynEditPrint1, 'TSynEditPrint', iptrw);
17598:     Printout1, 'TMenuItem', iptrw);
17599:     mnPrintColors1, 'TMenuItem', iptrw);
17600:     dlgFilePrint, 'TPrintDialog', iptrw);
17601:     dlgPrintFont1, 'TFontDialog', iptrw);
17602:     mnuPrintFont1, 'TMenuItem', iptrw);
17603:     Include1, 'TMenuItem', iptrw);
17604:     CodeCompletionList1, 'TMenuItem', iptrw);
17605:     IncludeList1, 'TMenuItem', iptrw);
17606:     ImageList1, 'TImageList', iptrw);
17607:     ImageList2, 'TImageList', iptrw);
17608:     CoolBar1, 'TCoolBar', iptrw);
17609:     ToolBar1, 'TToolBar', iptrw);
17610:     tbtnLoad, 'TToolButton', iptrw);
17611:     ToolButton2, 'TToolButton', iptrw);
17612:     tbtnFind, 'TToolButton', iptrw);
17613:     tbtnCompile, 'TToolButton', iptrw);
17614:     tbtnTrans, 'TToolButton', iptrw);
17615:     tbtnUseCase, 'TToolButton', iptrw); //3.8
17616:     toolbtnTutorial, 'TToolButton', iptrw);
17617:     tbtn6res, 'TToolButton', iptrw);
17618:     ToolButton5, 'TToolButton', iptrw);
17619:     ToolButton1, 'TToolButton', iptrw);
17620:     ToolButton3, 'TToolButton', iptrw);
17621:     statusBar1, 'TStatusBar', iptrw);

```



```

17622: SaveOutput1', 'TMenuItem', iptrw);
17623: ExportClipboard1', 'TMenuItem', iptrw);
17624: Close1', 'TMenuItem', iptrw);
17625: Manual1', 'TMenuItem', iptrw);
17626: About2', 'TMenuItem', iptrw);
17627: loadLastfile1', 'TMenuItem', iptrw);
17628: imglogo', 'TImage', iptrw);
17629: cedebug', 'TPSScriptDebugger', iptrw);
17630: debugPopupMenu1', 'TPopupMenu', iptrw);
17631: BreakPointMenu', 'TMenuItem', iptrw);
17632: Decompile1', 'TMenuItem', iptrw);
17633: StepInto1', 'TMenuItem', iptrw);
17634: StepOut1', 'TMenuItem', iptrw);
17635: Reset1', 'TMenuItem', iptrw);
17636: DebugRun1', 'TMenuItem', iptrw);
17637: PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
17638: PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
17639: PSImport_Forms1', 'TPSImport_Forms', iptrw);
17640: PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
17641: tutorial4', 'TMenuItem', iptrw);
17642: ExporttoClipboard1', 'TMenuItem', iptrw);
17643: ImportfromClipboard1', 'TMenuItem', iptrw);
17644: N4', 'TMenuItem', iptrw);
17645: N5', 'TMenuItem', iptrw);
17646: N6', 'TMenuItem', iptrw);
17647: ImportfromClipboard2', 'TMenuItem', iptrw);
17648: tutorial1', 'TMenuItem', iptrw);
17649: N7', 'TMenuItem', iptrw);
17650: ShowSpecChars1', 'TMenuItem', iptrw);
17651: OpenDirectory1', 'TMenuItem', iptrw);
17652: procMess', 'TMenuItem', iptrw);
17653: tbbtnUseCase', 'TToolButton', iptrw);
17654: ToolButton7', 'TToolButton', iptrw);
17655: EditFont1', 'TMenuItem', iptrw);
17656: UseCasel', 'TMenuItem', iptrw);
17657: tutorial21', 'TMenuItem', iptrw);
17658: OpenUseCasel', 'TMenuItem', iptrw);
17659: PSImport_DBI', 'TPSImport_DB', iptrw);
17660: tutorial31', 'TMenuItem', iptrw);
17661: SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
17662: HTMLSyntax1', 'TMenuItem', iptrw);
17663: ShowInterfaces1', 'TMenuItem', iptrw);
17664: Tutorial5', 'TMenuItem', iptrw);
17665: AllFunctionsList1', 'TMenuItem', iptrw);
17666: ShowLastException1', 'TMenuItem', iptrw);
17667: PlayMP31', 'TMenuItem', iptrw);
17668: SynTeXSyn1', 'TSynTeXSyn', iptrw);
17669: texSyntax1', 'TMenuItem', iptrw);
17670: N8', 'TMenuItem', iptrw);
17671: GetEmails1', 'TMenuItem', iptrw);
17672: SynCppSyn1', 'TSynCppSyn', iptrw);
17673: CSyntax1', 'TMenuItem', iptrw);
17674: Tutorial6', 'TMenuItem', iptrw);
17675: New1', 'TMenuItem', iptrw);
17676: AllObjectsList1', 'TMenuItem', iptrw);
17677: LoadBytecode1', 'TMenuItem', iptrw);
17678: CipherFile1', 'TMenuItem', iptrw);
17679: N9', 'TMenuItem', iptrw);
17680: N10', 'TMenuItem', iptrw);
17681: Tutorial11', 'TMenuItem', iptrw);
17682: Tutorial71', 'TMenuItem', iptrw);
17683: UpdateService1', 'TMenuItem', iptrw);
17684: PascalSchool1', 'TMenuItem', iptrw);
17685: Tutorial81', 'TMenuItem', iptrw);
17686: DelphiSite1', 'TMenuItem', iptrw);
17687: Output1', 'TMenuItem', iptrw);
17688: TerminalStyle1', 'TMenuItem', iptrw);
17689: ReadOnly1', 'TMenuItem', iptrw);
17690: ShellStyle1', 'TMenuItem', iptrw);
17691: BigScreen1', 'TMenuItem', iptrw);
17692: Tutorial91', 'TMenuItem', iptrw);
17693: SaveOutput2', 'TMenuItem', iptrw);
17694: N11', 'TMenuItem', iptrw);
17695: SaveScreenshot', 'TMenuItem', iptrw);
17696: Tutorial101', 'TMenuItem', iptrw);
17697: SQLSyntax1', 'TMenuItem', iptrw);
17698: SynSQLSyn1', 'TSynSQLSyn', iptrw);
17699: Console1', 'TMenuItem', iptrw);
17700: SynXMLSyn1', 'TSynXMLSyn', iptrw);
17701: XMLSyntax1', 'TMenuItem', iptrw);
17702: ComponentCount1', 'TMenuItem', iptrw);
17703: NewInstancel', 'TMenuItem', iptrw);
17704: toolbtnTutorial', 'TToolButton', iptrw);
17705: Memory1', 'TMenuItem', iptrw);
17706: SynJavaSyn1', 'TSynJavaSyn', iptrw);
17707: JavaSyntax1', 'TMenuItem', iptrw);
17708: SyntaxCheck1', 'TMenuItem', iptrw);
17709: Tutorial10Statistics1', 'TMenuItem', iptrw);
17710: ScriptExplorer1', 'TMenuItem', iptrw);

```

```
17711: FormOutput1', 'TMenuItem', iptrw);
17712: ArduinoDump1', 'TMenuItem', iptrw);
17713: AndroidDump1', 'TMenuItem', iptrw);
17714: GotoEnd1', 'TMenuItem', iptrw);
17715: AllResourceList1', 'TMenuItem', iptrw);
17716: ToolButton4', 'TToolButton', iptrw);
17717: tbtn6res', 'TToolButton', iptrw);
17718: Tutorial11Forms1', 'TMenuItem', iptrw);
17719: Tutorial12SQL1', 'TMenuItem', iptrw);
17720: ResourceExplorel', 'TMenuItem', iptrw);
17721: Infol', 'TMenuItem', iptrw);
17722: N12', 'TMenuItem', iptrw);
17723: CryptoBox1', 'TMenuItem', iptrw);
17724: Tutorial13Ciphering1', 'TMenuItem', iptrw);
17725: CipherFile2', 'TMenuItem', iptrw);
17726: N13', 'TMenuItem', iptrw);
17727: ModulesCount1', 'TMenuItem', iptrw);
17728: AddOns2', 'TMenuItem', iptrw);
17729: N4GewinntGame1', 'TMenuItem', iptrw);
17730: DocuforAddOns1', 'TMenuItem', iptrw);
17731: Tutorial14Async1', 'TMenuItem', iptrw);
17732: Lessons15Review1', 'TMenuItem', iptrw);
17733: SynPHPSyn1', 'TSynPHPSyn', iptrw);
17734: PHPSyntax1', 'TMenuItem', iptrw);
17735: Breakpoint1', 'TMenuItem', iptrw);
17736: SerialRS2321', 'TMenuItem', iptrw);
17737: N14', 'TMenuItem', iptrw);
17738: SynCSSyn1', 'TSynCSSyn', iptrw);
17739: CSyntax2', 'TMenuItem', iptrw);
17740: Calculator1', 'TMenuItem', iptrw);
17741: tbtnSerial', 'TToolButton', iptrw);
17742: ToolButton8', 'TToolButton', iptrw);
17743: Tutorial151', 'TMenuItem', iptrw);
17744: N15', 'TMenuItem', iptrw);
17745: N16', 'TMenuItem', iptrw);
17746: ControlBar1', 'TControlBar', iptrw);
17747: ToolBar2', 'TToolBar', iptrw);
17748: BtnOpen', 'TToolButton', iptrw);
17749: BtnSave', 'TToolButton', iptrw);
17750: BtnPrint', 'TToolButton', iptrw);
17751: BtnColors', 'TToolButton', iptrw);
17752: btnClassReport', 'TToolButton', iptrw);
17753: BtnRotateRight', 'TToolButton', iptrw);
17754: BtnFullSize', 'TToolButton', iptrw);
17755: BtnFitToWindowSize', 'TToolButton', iptrw);
17756: BtnZoomMinus', 'TToolButton', iptrw);
17757: BtnZoomPlus', 'TToolButton', iptrw);
17758: Panell1', 'TPanel', iptrw);
17759: LabelBrettgroesse', 'TLabel', iptrw);
17760: CB1SCList', 'TComboBox', iptrw);
17761: ImageListNormal', 'TImageList', iptrw);
17762: spbbtnexplore', 'TSpeedButton', iptrw);
17763: spbbtnexample', 'TSpeedButton', iptrw);
17764: spbsaveas', 'TSpeedButton', iptrw);
17765: imglogobox', 'TImage', iptrw);
17766: EnlargeFont1', 'TMenuItem', iptrw);
17767: EnlargeFont2', 'TMenuItem', iptrw);
17768: ShrinkFont1', 'TMenuItem', iptrw);
17769: ThreadDemol', 'TMenuItem', iptrw);
17770: HEXEditor1', 'TMenuItem', iptrw);
17771: HEXView1', 'TMenuItem', iptrw);
17772: HEXInspect1', 'TMenuItem', iptrw);
17773: SynExporterHTML1', 'TSynExporterHTML', iptrw);
17774: ExporttoHTML1', 'TMenuItem', iptrw);
17775: ClassCount1', 'TMenuItem', iptrw);
17776: HTMLOutput1', 'TMenuItem', iptrw);
17777: HEXEditor2', 'TMenuItem', iptrw);
17778: Minesweeper1', 'TMenuItem', iptrw);
17779: N17', 'TMenuItem', iptrw);
17780: PicturePuzzle1', 'TMenuItem', iptrw);
17781: sbvclhelp', 'TSpeedButton', iptrw);
17782: DependencyWalker1', 'TMenuItem', iptrw);
17783: WebScanner1', 'TMenuItem', iptrw);
17784: View1', 'TMenuItem', iptrw);
17785: mnToolbar1', 'TMenuItem', iptrw);
17786: mnStatusbar2', 'TMenuItem', iptrw);
17787: mnConsole2', 'TMenuItem', iptrw);
17788: mnCoolbar2', 'TMenuItem', iptrw);
17789: mnSplitter2', 'TMenuItem', iptrw);
17790: WebServer1', 'TMenuItem', iptrw);
17791: Tutorial17Server1', 'TMenuItem', iptrw);
17792: Tutorial18Arduinol', 'TMenuItem', iptrw);
17793: SynPerlSyn1', 'TSynPerlSyn', iptrw);
17794: PerlSyntax1', 'TMenuItem', iptrw);
17795: SynPythonSyn1', 'TSynPythonSyn', iptrw);
17796: PythonSyntax1', 'TMenuItem', iptrw);
17797: DMathLibrary1', 'TMenuItem', iptrw);
17798: IntfNavigator1', 'TMenuItem', iptrw);
17799: EnlargeFontConsole1', 'TMenuItem', iptrw);
```

```

17800: ShrinkFontConsole1', 'TMenuItem', iptrw);
17801: SetInterfaceList1', 'TMenuItem', iptrw);
17802: popintfList', 'TPopupMenu', iptrw);
17803: intfAdd1', 'TMenuItem', iptrw);
17804: intfDelete1', 'TMenuItem', iptrw);
17805: intfRefactor1', 'TMenuItem', iptrw);
17806: Defactor1', 'TMenuItem', iptrw);
17807: Tutorial19COMArduinol', 'TMenuItem', iptrw);
17808: Tutorial20Regex', 'TMenuItem', iptrw);
17809: N18', 'TMenuItem', iptrw);
17810: ManualE1', 'TMenuItem', iptrw);
17811: FullTextFinder1', 'TMenuItem', iptrw);
17812: Move1', 'TMenuItem', iptrw);
17813: FractalDemol', 'TMenuItem', iptrw);
17814: Tutorial21Android1', 'TMenuItem', iptrw);
17815: Tutorial0Function1', 'TMenuItem', iptrw);
17816: SimuLogBox1', 'TMenuItem', iptrw);
17817: OpenExamples1', 'TMenuItem', iptrw);
17818: SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
17819: JavaScriptSyntax1', 'TMenuItem', iptrw);
17820: Halt1', 'TMenuItem', iptrw);
17821: CodeSearch1', 'TMenuItem', iptrw);
17822: SynRubySyn1', 'TSynRubySyn', iptrw);
17823: RubySyntax1', 'TMenuItem', iptrw);
17824: Undo1', 'TMenuItem', iptrw);
17825: SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
17826: LinuxShellScript1', 'TMenuItem', iptrw);
17827: Rename1', 'TMenuItem', iptrw);
17828: spdcodesearch', 'TSpeedButton', iptrw);
17829: Preview1', 'TMenuItem', iptrw);
17830: Tutorial22Services1', 'TMenuItem', iptrw);
17831: Tutorial23RealTime1', 'TMenuItem', iptrw);
17832: Configuration1', 'TMenuItem', iptrw);
17833: MP3Player1', 'TMenuItem', iptrw);
17834: DLLSpy1', 'TMenuItem', iptrw);
17835: SynURIOpener1', 'TSynURIOpener', iptrw);
17836: SynURISyn1', 'TSynURISyn', iptrw);
17837: URILinksClicks1', 'TMenuItem', iptrw);
17838: EditReplac1', 'TMenuItem', iptrw);
17839: GotoLine1', 'TMenuItem', iptrw);
17840: ActiveLineColor1', 'TMenuItem', iptrw);
17841: ConfigFile1', 'TMenuItem', iptrw);
17842: Sort1IntfList', 'TMenuItem', iptrw);
17843: Redol', 'TMenuItem', iptrw);
17844: Tutorial24CleanCode1', 'TMenuItem', iptrw);
17845: Tutorial25Configuration1', 'TMenuItem', iptrw);
17846: IndentSelection1', 'TMenuItem', iptrw);
17847: UnindentSection1', 'TMenuItem', iptrw);
17848: SkyStyle1', 'TMenuItem', iptrw);
17849: N19', 'TMenuItem', iptrw);
17850: CountWords1', 'TMenuItem', iptrw);
17851: imbookmarkimages', 'TImageList', iptrw);
17852: Bookmark11', 'TMenuItem', iptrw);
17853: N20', 'TMenuItem', iptrw);
17854: Bookmark21', 'TMenuItem', iptrw);
17855: Bookmark31', 'TMenuItem', iptrw);
17856: Bookmark41', 'TMenuItem', iptrw);
17857: SynMultiSyn1', 'TSynMultiSyn', iptrw);
17858:
17859: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
17860: Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
17861: Procedure PSScriptCompile( Sender : TPSScript)
17862: Procedure Compile1Click( Sender : TObject)
17863: Procedure PSScriptExecute( Sender : TPSScript)
17864: Procedure open1Click( Sender : TObject)
17865: Procedure Save2Click( Sender : TObject)
17866: Procedure Savebefore1Click( Sender : TObject)
17867: Procedure Largefont1Click( Sender : TObject)
17868: Procedure FormActivate( Sender : TObject)
17869: Procedure SBytecode1Click( Sender : TObject)
17870: Procedure FormKeyPress( Sender : TObject; var Key : Char)
17871: Procedure Saveas3Click( Sender : TObject)
17872: Procedure Clear1Click( Sender : TObject)
17873: Procedure Slinenumbers1Click( Sender : TObject)
17874: Procedure About1Click( Sender : TObject)
17875: Procedure Search1Click( Sender : TObject)
17876: Procedure FormCreate( Sender : TObject)
17877: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
17878: var Action : TSynReplaceAction)
17879: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
17880: Procedure WordWrap1Click( Sender : TObject)
17881: Procedure SearchNext1Click( Sender : TObject)
17882: Procedure Replace1Click( Sender : TObject)
17883: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
17884: Procedure ShowInclude1Click( Sender : TObject)
17885: Procedure Printout1Click( Sender : TObject)
17886: Procedure mnuPrintFont1Click( Sender : TObject)
17887: Procedure Include1Click( Sender : TObject)
17888: Procedure FormDestroy( Sender : TObject)

```

```

17889: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
17890: Procedure UpdateView1Click( Sender : TObject)
17891: Procedure CodeCompletionList1Click( Sender : TObject)
17892: Procedure SaveOutput1Click( Sender : TObject)
17893: Procedure ExportClipboard1Click( Sender : TObject)
17894: Procedure Close1Click( Sender : TObject)
17895: Procedure Manual1Click( Sender : TObject)
17896: Procedure LoadLastFile1Click( Sender : TObject)
17897: Procedure Memo1Change( Sender : TObject)
17898: Procedure Decompile1Click( Sender : TObject)
17899: Procedure StepInto1Click( Sender : TObject)
17900: Procedure StepOut1Click( Sender : TObject)
17901: Procedure Reset1Click( Sender : TObject)
17902: Procedure cedebugAfterExecute( Sender : TPSScript)
17903: Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
17904: Procedure cedebugCompile( Sender : TPSScript)
17905: Procedure cedebugExecute( Sender : TPSScript)
17906: Procedure cedebugIdle( Sender : TObject)
17907: Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
17908: Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
17909: Procedure BreakPointMenuClick( Sender : TObject)
17910: Procedure DebugRun1Click( Sender : TObject)
17911: Procedure tutorial4Click( Sender : TObject)
17912: Procedure ImportfromClipboard1Click( Sender : TObject)
17913: Procedure ImportfromClipboard2Click( Sender : TObject)
17914: Procedure tutorial1Click( Sender : TObject)
17915: Procedure ShowSpecChars1Click( Sender : TObject)
17916: Procedure StatusBar1Db1Click( Sender : TObject)
17917: Procedure PSScriptLine( Sender : TObject)
17918: Procedure OpenDirectory1Click( Sender : TObject)
17919: Procedure procMessClick( Sender : TObject)
17920: Procedure tbtnUseCaseClick( Sender : TObject)
17921: Procedure EditFont1Click( Sender : TObject)
17922: Procedure tutorial21Click( Sender : TObject)
17923: Procedure tutorial31Click( Sender : TObject)
17924: Procedure HTMLSyntax1Click( Sender : TObject)
17925: Procedure ShowInterfaces1Click( Sender : TObject)
17926: Procedure Tutorial5Click( Sender : TObject)
17927: Procedure ShowLastException1Click( Sender : TObject)
17928: Procedure PlayMP31Click( Sender : TObject)
17929: Procedure AllFunctionsList1Click( Sender : TObject)
17930: Procedure texSyntax1Click( Sender : TObject)
17931: Procedure GetEMails1Click( Sender : TObject)
17932: procedure DelphiSite1Click(Sender: TObject);
17933: procedure TerminalStyle1Click(Sender: TObject);
17934: procedure ReadOnly1Click(Sender: TObject);
17935: procedure ShellStyle1Click(Sender: TObject);
17936: procedure Console1Click(Sender: TObject); //3.2
17937: procedure BigScreen1Click(Sender: TObject);
17938: procedure Tutorial91Click(Sender: TObject);
17939: procedure SaveScreenshotClick(Sender: TObject);
17940: procedure Tutorial101Click(Sender: TObject);
17941: procedure SQLSyntax1Click(Sender: TObject);
17942: procedure XMLSyntax1Click(Sender: TObject);
17943: procedure ComponentCount1Click(Sender: TObject);
17944: procedure NewInstance1Click(Sender: TObject);
17945: procedure CSyntax1Click(Sender: TObject);
17946: procedure Tutorial6Click(Sender: TObject);
17947: procedure New1Click(Sender: TObject);
17948: procedure AllObjectsList1Click(Sender: TObject);
17949: procedure LoadBytecode1Click(Sender: TObject);
17950: procedure CipherFile1Click(Sender: TObject); //V3.5
17951: procedure NewInstance1Click(Sender: TObject);
17952: procedure toolbtnTutorialClick(Sender: TObject);
17953: procedure Memory1Click(Sender: TObject);
17954: procedure JavaSyntax1Click(Sender: TObject);
17955: procedure SyntaxCheck1Click(Sender: TObject);
17956: procedure ScriptExplorer1Click(Sender: TObject);
17957: procedure FormOutput1Click(Sender: TObject); //V3.6
17958: procedure GotoEnd1Click(Sender: TObject);
17959: procedure AllResourceList1Click(Sender: TObject);
17960: procedure tbtn6resClick(Sender: TObject); //V3.7
17961: procedure Info1Click(Sender: TObject);
17962: procedure Tutorial10Statistics1Click(Sender: TObject);
17963: procedure Tutorial11Forms1Click(Sender: TObject);
17964: procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
17965: procedure ResourceExplore1Click(Sender: TObject);
17966: procedure Info1Click(Sender: TObject);
17967: procedure CryptoBox1Click(Sender: TObject);
17968: procedure ModulesCount1Click(Sender: TObject);
17969: procedure N4GewinntGame1Click(Sender: TObject);
17970: procedure PHPSyntax1Click(Sender: TObject);
17971: procedure SerialRS2321Click(Sender: TObject);
17972: procedure CSyntax2Click(Sender: TObject);
17973: procedure Calculator1Click(Sender: TObject);
17974: procedure Tutorial13Ciphering1Click(Sender: TObject);
17975: procedure Tutorial14Async1Click(Sender: TObject);
17976: procedure PHPSyntax1Click(Sender: TObject);
17977: procedure BtnZoomPlusClick(Sender: TObject);

```



```

17978: procedure BtnZoomMinusClick(Sender: TObject);
17979: procedure btnClassReportClick(Sender: TObject);
17980: procedure ThreadDemolClick(Sender: TObject);
17981: procedure HEXView1Click(Sender: TObject);
17982: procedure ExporttoHTML1Click(Sender: TObject);
17983: procedure Minesweeper1Click(Sender: TObject);
17984: procedure PicturePuzzle1Click(Sender: TObject); //V3.9
17985: procedure sbvclhelpClick(Sender: TObject);
17986: procedure DependencyWalker1Click(Sender: TObject);
17987: procedure CB1SCListDrawItem(Control: TWinControl; Index: Int; aRect: TRect; State: TOwnerDrawState);
17988: procedure WebScanner1Click(Sender: TObject);
17989: procedure mnToolbar1Click(Sender: TObject);
17990: procedure mnStatusbar2Click(Sender: TObject);
17991: procedure mnConsole2Click(Sender: TObject);
17992: procedure mnCoolbar2Click(Sender: TObject);
17993: procedure mnSplitter2Click(Sender: TObject);
17994: procedure WebServer1Click(Sender: TObject);
17995: procedure PerlSyntax1Click(Sender: TObject);
17996: procedure PythonSyntax1Click(Sender: TObject);
17997: procedure DMathLibrary1Click(Sender: TObject);
17998: procedure IntfNavigator1Click(Sender: TObject);
17999: procedure FullTextFinder1Click(Sender: TObject);
18000: function AppName: string;
18001: function ScriptName: string;
18002: function LastName: string;
18003: procedure FractalDemolClick(Sender: TObject);
18004: procedure SimuLogBox1Click(Sender: TObject);
18005: procedure OpenExamples1Click(Sender: TObject);
18006: procedure Halt1Click(Sender: TObject);
18007: procedure Stop;
18008: procedure CodeSearch1Click(Sender: TObject);
18009: procedure RubySyntax1Click(Sender: TObject);
18010: procedure Undo1Click(Sender: TObject);
18011: procedure LinuxShellScript1Click(Sender: TObject);
18012: procedure WebScannerDirect(urls: string);
18013: procedure WebScanner(urls: string);
18014: procedure LoadInterfaceList2;
18015: procedure DLLSpy1Click(Sender: TObject);
18016: procedure Memo1Db1Click(Sender: TObject);
18017: procedure URILinksClicks1Click(Sender: TObject);
18018: procedure GotoLine1Click(Sender: TObject);
18019: procedure ConfigFile1Click(Sender: TObject);
18020: procedure Sort1IntflistClick(Sender: TObject);
18021: procedure RedolClick(Sender: TObject);
18022: procedure Tutorial24CleanCode1Click(Sender: TObject);
18023: procedure IndentSelection1Click(Sender: TObject);
18024: procedure UnindentSection1Click(Sender: TObject);
18025: procedure SkyStyle1Click(Sender: TObject);
18026: procedure CountWords1Click(Sender: TObject);
18027: procedure Memo1PlaceBookmark(Sender: TObject; var Mark: TSynEditMark);
18028: procedure Memo1GutterClick(Sender: TObject; Button: TMouseButton; X, Y, Line: Integer; Mark: TSynEditMark);
18029: procedure Bookmark11Click(Sender: TObject);
18030: procedure Bookmark21Click(Sender: TObject);
18031: procedure Bookmark31Click(Sender: TObject);
18032: procedure Bookmark41Click(Sender: TObject);
18033: procedure SynMultiSyn1CustomRange(Sender: TSynMultiSyn; Operation: TRangeOperation; var Range: Pointer);
18034: 'STATMemoryReport', 'boolean', iptrw);
18035: 'IPPort', 'integer', iptrw);
18036: 'COMPort', 'integer', iptrw);
18037: 'lbintflist', 'TListBox', iptrw);
18038: Function GetStatChange: boolean
18039: Procedure SetStatChange(vstat: boolean)
18040: Function GetActFileName: string
18041: Procedure SetActFileName(vname: string)
18042: Function GetLastFileName: string
18043: Procedure SetLastFileName(vname: string)
18044: Procedure WebScannerDirect(urls: string)
18045: Procedure LoadInterfaceList2
18046: Function GetStatExecuteShell: boolean
18047: Procedure DoEditorExecuteCommand(EditorCommand: word)
18048: function GetActiveLineColor: TColor
18049: procedure SetActiveLineColor(acolor: TColor)
18050: procedure ScriptListBox1Click(Sender: TObject);
18051: procedure Memo2KeyPress(Sender: TObject; var Key: Char);
18052: procedure EnlargeGutter1Click(Sender: TObject);
18053: procedure Tetris1Click(Sender: TObject);
18054: procedure ToDoList1Click(Sender: TObject);
18055: procedure ProcessList1Click(Sender: TObject);
18056: procedure MetricReport1Click(Sender: TObject);
18057: procedure ProcessList1Click(Sender: TObject);
18058: procedure TCPSockets1Click(Sender: TObject);
18059: procedure ConfigUpdate1Click(Sender: TObject);
18060: procedure ADOWorkbench1Click(Sender: TObject);
18061: procedure SocketServer1Click(Sender: TObject);
18062: procedure FormDemolClick(Sender: TObject);
18063: procedure Richedit1Click(Sender: TObject);
18064: procedure SimpleBrowser1Click(Sender: TObject);
18065: procedure DOSShell1Click(Sender: TObject);
18066: procedure SynExport1Click(Sender: TObject);

```

```

18067: procedure ExporttoRTF1Click(Sender: TObject);
18068: procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
18069: procedure SOAPTester1Click(Sender: TObject);
18070: procedure Sniffer1Click(Sender: TObject);
18071: procedure AutoDetectSyntax1Click(Sender: TObject);
18072: procedure FPlot1Click(Sender: TObject);
18073: procedure PasStyle1Click(Sender: TObject);
18074: procedure Tutorial183RGBLED1Click(Sender: TObject);
18075: procedure ReversilClick(Sender: TObject);
18076: procedure ManualmaXbox1Click(Sender: TObject);
18077: procedure BlaisePascalMagazine1Click(Sender: TObject);
18078: procedure AddToDo1Click(Sender: TObject);
18079: procedure CreateGUID1Click(Sender: TObject);
18080: procedure Tutorial27XML1Click(Sender: TObject);
18081: procedure CreateDLLStub1Click(Sender: TObject);
18082: procedure Tutorial28DLL1Click(Sender: TObject);
18083: procedure ResetKeyPressed;
18084: procedure FileChanges1Click(Sender: TObject);
18085: procedure OpenGLTry1Click(Sender: TObject);
18086: procedure AllUnitList1Click(Sender: TObject);
18087:
18088:
18089: //-----
18090: //*****mX4 Editor SynEdit Tools API *****
18091: //-----
18092: (*-----*)
18093: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
18094: begin
18095:   //with RegClassS(CL, 'TCustomControl', 'TCustomSynEdit') do
18096:   with FindClass('TCustomControl', 'TCustomSynEdit') do begin
18097:     Constructor Create( AOwner : TComponent)
18098:     SelStart, 'Integer', iptrw);
18099:     SelEnd, 'Integer', iptrw); AlwaysShowCaret, 'Boolean', iptrw);
18100:     Procedure UpdateCaret
18101:     Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1: TShiftState; Key2: word; SS2: TShiftState);
18102:     Procedure AddKey(Command: TSynEditorCommand; Key1: word; SS1: TShiftState; Key2: word; SS2: TShiftState);
18103:     Procedure BeginUndoBlock
18104:     Procedure BeginUpdate
18105:     Function CaretInView : Boolean
18106:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
18107:     Procedure Clear
18108:     Procedure ClearAll
18109:     Procedure ClearBookMark( BookMark : Integer)
18110:     Procedure ClearSelection
18111:     CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
18112:     Procedure ClearUndo
18113:     Procedure CopyToClipboard
18114:     Procedure CutToClipboard
18115:     Procedure DoCopyToClipboard( const SText : string)
18116:     Procedure EndUndoBlock
18117:     Procedure EndUpdate
18118:     Procedure EnsureCursorPosVisible
18119:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
18120:     Procedure FindMatchingBracket
18121:     Function GetMatchingBracket : TBufferCoord
18122:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
18123:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
18124:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
18125:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
18126:       : TSynHighlighterAttributes) : boolean
18127:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
18128:       var TokenType, Start : integer; var Attri: TSynHighlighterAttributes):boolean
18129:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
18130:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
18131:     Procedure GotoBookMark( BookMark : Integer)
18132:     Procedure GotoLineAndCenter( ALine : Integer)
18133:     Function IdentChars : TSynIdentChars
18134:     Procedure InvalidateGutter
18135:     Procedure InvalidateGutterLine( aLine : integer)
18136:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
18137:     Procedure InvalidateLine( Line : integer)
18138:     Procedure InvalidateLines( FirstLine, LastLine : integer)
18139:     Procedure InvalidateSelection
18140:     Function IsBookmark( BookMark : integer) : boolean
18141:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
18142:     Procedure LockUndo
18143:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
18144:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
18145:     Function LineToRow( aLine : integer) : integer
18146:     Function RowToLine( aRow : integer) : integer
18147:     Function NextWordPos : TBufferCoord
18148:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
18149:     Procedure PasteFromClipboard
18150:     Function WordStart : TBufferCoord
18151:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
18152:     Function WordEnd : TBufferCoord
18153:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
18154:     Function PrevWordPos : TBufferCoord
18155:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord

```

```

18156: Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
18157: Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
18158: Procedure Redo
18159: Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer));
18160: Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
18161: Function RowColToCharIndex( RowCol : TBufferCoord) : integer
18162: Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
18163: Procedure SelectAll
18164: Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
18165: Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
18166: Procedure SetDefaultKeystrokes
18167: Procedure SetSelWord
18168: Procedure SetWordBlock( Value : TBufferCoord)
18169: Procedure Undo
18170: Procedure UnlockUndo
18171: Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
18172: Procedure AddKeyUpHandler( aHandler : TKeyEvent)
18173: Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
18174: Procedure AddKeyDownHandler( aHandler : TKeyEvent)
18175: Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
18176: Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
18177: Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
18178: Procedure AddFocusControl( aControl : TWinControl)
18179: Procedure RemoveFocusControl( aControl : TWinControl)
18180: Procedure AddMouseDownHandler( aHandler : TMouseEvent)
18181: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
18182: Procedure AddMouseUpHandler( aHandler : TMouseEvent)
18183: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
18184: Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
18185: Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
18186: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
18187: Procedure RemoveLinesPointer
18188: Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
18189: Procedure UnHookTextBuffer
18190: BlockBegin', 'TBufferCoord', iptrw);
18191: BlockEnd', 'TBufferCoord', iptrw);
18192: CanPaste', 'Boolean', iptr);
18193: CanRedo', 'boolean', iptr);
18194: CanUndo', 'boolean', iptr);
18195: CaretX', 'Integer', iptrw);
18196: CaretY', 'Integer', iptrw);
18197: CaretXY', 'TBufferCoord', iptrw);
18198: ActiveLineColor', 'TColor', iptrw);
18199: DisplayX', 'Integer', iptr);
18200: DisplayY', 'Integer', iptr);
18201: DisplayXY', 'TDisplayCoord', iptr);
18202: DisplayLineCount', 'integer', iptr);
18203: CharsInWindow', 'Integer', iptr);
18204: CharWidth', 'integer', iptr);
18205: Font', 'TFont', iptrw);
18206: GutterWidth', 'Integer', iptr);
18207: Highlighter', 'TSynCustomHighlighter', iptrw);
18208: LeftChar', 'Integer', iptrw);
18209: LineHeight', 'integer', iptr);
18210: LinesInWindow', 'Integer', iptr);
18211: LineText', 'string', iptrw);
18212: Lines', 'TStrings', iptrw);
18213: Marks', 'TSynEditMarkList', iptr);
18214: MaxScrollWidth', 'integer', iptrw);
18215: Modified', 'Boolean', iptrw);
18216: PaintLock', 'Integer', iptr);
18217: ReadOnly', 'Boolean', iptrw);
18218: SearchEngine', 'TSynEditSearchCustom', iptrw);
18219: SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
18220: SelTabBlock', 'Boolean', iptr);
18221: SelTabLine', 'Boolean', iptr);
18222: SelText', 'string', iptrw);
18223: StateFlags', 'TSynStateFlags', iptr);
18224: Text', 'string', iptrw);
18225: TopLine', 'Integer', iptrw);
18226: WordAtCursor', 'string', iptr);
18227: WordAtMouse', 'string', iptr);
18228: UndoList', 'TSynEditUndoList', iptr);
18229: RedoList', 'TSynEditUndoList', iptr);
18230: OnProcessCommand', 'TProcessCommandEvent', iptrw);
18231: BookMarkOptions', 'TSynBookMarkOpt', iptrw);
18232: BorderStyle', 'TSynBorderStyle', iptrw);
18233: ExtraLineSpacing', 'integer', iptrw);
18234: Gutter', 'TSynGutter', iptrw);
18235: HideSelection', 'boolean', iptrw);
18236: InsertCaret', 'TSynEditCaretType', iptrw);
18237: InsertMode', 'boolean', iptrw);
18238: IsScrolling', 'Boolean', iptr);
18239: Keystrokes', 'TSynEditKeyStrokes', iptrw);
18240: MaxUndo', 'Integer', iptrw);
18241: Options', 'TSynEditorOptions', iptrw);
18242: OverwriteCaret', 'TSynEditCaretType', iptrw);
18243: RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
18244: ScrollHintColor', 'TColor', iptrw);

```

```

18245: ScrollHintFormat', 'TScrollHintFormat', iptrw);
18246: ScrollBars', 'TScrollStyle', iptrw);
18247: SelectedColor', 'TSynSelectedColor', iptrw);
18248: SelectionMode', 'TSynSelectionMode', iptrw);
18249: ActiveSelectionMode', 'TSynSelectionMode', iptrw);
18250: TabWidth', 'integer', iptrw); WantReturns', 'boolean', iptrw);
18251: WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
18252: WordWrapGlyph', 'TSynGlyph', iptrw);
18253: OnChange', 'TNotifyEvent', iptrw);
18254: OnClearBookmark', 'TPlaceMarkEvent', iptrw);
18255: OnCommandProcessed', 'TProcessCommandEvent', iptrw);
18256: OnContextHelp', 'TContextHelpEvent', iptrw);
18257: OnDropFiles', 'TDropFilesEvent', iptrw);
18258: OnGutterClick', 'TGutterClickEvent', iptrw);
18259: OnGutterGetText', 'TGutterGetTextEvent', iptrw);
18260: OnGutterPaint', 'TGutterPaintEvent', iptrw);
18261: OnMouseCursor', 'TMouseCursorEvent', iptrw);
18262: OnPaint', 'TPaintEvent', iptrw);
18263: OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
18264: OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
18265: OnReplaceText', 'TReplaceTextEvent', iptrw);
18266: OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
18267: OnStatusChange', 'TStatusChangeEvent', iptrw);
18268: OnPaintTransient', 'TPaintTransient', iptrw);
18269: OnScroll', 'TScrollEvent', iptrw);
18270: end;
18271: end;
18272: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
18273: Function GetPlaceableHighlighters : TSynHighlighterList
18274: Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
18275: Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
18276: Procedure GetEditorCommandValues( Proc : TGetStrProc)
18277: Procedure GetEditorCommandExtended( Proc : TGetStrProc)
18278: Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
18279: Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
18280: Function ConvertCodeStringToExtended( AString : String) : String
18281: Function ConvertExtendedToCodeString( AString : String) : String
18282: Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
18283: Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
18284: Function IndexToEditorCommand( const AIndex : Integer) : Integer
18285:
18286: TSynEditorOption = (
18287:   eoAltSetsColumnMode,           //Holding down the Alt Key will put the selection mode into columnar format
18288:   eoAutoIndent,                 //Will indent caret on newlines with same amount of leading whitespace as
18289:                                 // preceding line
18290:   eoAutoSizeMaxScrollWidth,     //Automatically resizes the MaxScrollWidth property when inserting text
18291:   eoDisableScrollArrows,        //Disables the scroll bar arrow buttons when you can't scroll in that
18292:                                 //direction any more
18293:   eoDragDropEditing,            //Allows to select a block of text and drag it within document to another
18294:                                 // location
18295:   eoDropFiles,                 //Allows the editor accept OLE file drops
18296:   eoEnhanceHomeKey,             //enhances home key positioning, similar to visual studio
18297:   eoEnhanceEndKey,             //enhances End key positioning, similar to JDeveloper
18298:   eoGroupUndo,                 //When undoing/redoning actions, handle all continous changes the same kind
18299:                                 // in one call
18300:                                 //instead undoing/redoning each command separately
18301:   eoHalfPageScroll,            //When scrolling with page-up and page-down commands, only scroll a half
18302:                                 //page at a time
18303:   eoHideShowScrollbars,        //if enabled, then scrollbars will only show if necessary.
18304:   If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
18305:   eoKeepCaretX,                //When moving through lines w/o cursor Past EOL, keeps X position of cursor
18306:   eoNoCaret,                   //Makes it so the caret is never visible
18307:   eoNoSelection,               //Disables selecting text
18308:   eoRightMouseMovesCursor,     //When clicking with right mouse for popup menu, moves cursor to location
18309:   eoScrollByOneLess,           //Forces scrolling to be one less
18310:   eoScrollHintFollows,         //The scroll hint follows the mouse when scrolling vertically
18311:   eoScrollPastEof,             //Allows the cursor to go past the end of file marker
18312:   eoScrollPastEol,             //Allows cursor to go past last character into white space at end of a line
18313:   eoShowScrollHint,            //Shows a hint of the visible line numbers when scrolling vertically
18314:   eoShowSpecialChars,          //Shows the special Characters
18315:   eoSmartTabDelete,            //similar to Smart Tabs, but when you delete characters
18316:   eoSmartTabs,                 //When tabbing, cursor will go to non-white space character of previous line
18317:   eoSpecialLineDefaultFg,       //disables the foreground text color override using OnSpecialLineColor event
18318:   eoTabIndent,                 //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
18319:   eoTabsToSpaces,              //Converts a tab character to a specified number of space characters
18320:   eoTrimTrailingSpaces         //Spaces at the end of lines will be trimmed and not saved
18321:
18322:   *****Important Editor Short Cuts*****);
18323: Double click to select a word and count words with highlightning.
18324: Triple click to select a line.
18325: CTRL+SHIFT+click to extend a selection.
18326: Drag with the ALT key down to select columns of text !!!
18327: Drag and drop is supported.
18328: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
18329: Type CTRL+A to select all.
18330: Type CTRL+N to set a new line.
18331: Type CTRL+T to delete a line or token. //Tokenizer
18332: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
18333: Type CTRL+Shift+T to add ToDo in line and list.

```



```

18334:    Type CTRL+Shift+[0..9] to set bookmarks.    //Bookmark
18335:    Type CTRL[0..9] to jump or get to bookmarks.
18336:    Type Home to position cursor at beginning of current line and End to position it at end of line.
18337:    Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
18338:    Page Up and Page Down work as expected.
18339:    CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
18340:    using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
18341:
18342:    {$ Short Key Positions Ctrl<A-Z>: }
18343:    def
18344:        <A> Select All
18345:        <B> Count Words
18346:        <C> Copy
18347:        <D> Internet Start
18348:        <E> Script List
18349:        <F> Find
18350:        <G> Goto
18351:        <H> Mark Line
18352:        <I> Interface List
18353:        <J> Code Completion
18354:        <K> Console
18355:        <L> Interface List Box
18356:        <M> Font Larger -
18357:        <N> New Line
18358:        <O> Open File
18359:        <P> Font Smaller +
18360:        <Q> Quit
18361:        <R> Replace
18362:        <S> Save!
18363:        <T> Delete Line
18364:        <U> Use Case Editor
18365:        <V> Paste
18366:        <W> URI Links
18367:        <X> Reserved for coding use internal
18368:        <Y> Delete Line
18369:        <Z> Undo
18370:
18371:    ref
18372:        F1 Help
18373:        F2 Syntax Check
18374:        F3 Search Next
18375:        F4 New Instance
18376:        F5 Line Mark /Breakpoint
18377:        F6 Goto End
18378:        F7 Debug Step Into
18379:        F8 Debug Step Out
18380:        F9 Compile
18381:        F10 Menu
18382:        F11 Word Count Highlight
18383:        F12 Reserved for coding use internal
18384:
18385:    def ReservedWords: array[0..82] of string =
18386:        ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
18387:         'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
18388:         'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
18389:         'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
18390:         'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
18391:         'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
18392:         'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
18393:         'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
18394:         'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
18395:         'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
18396:         'public', 'published', 'def', 'ref', 'using', 'typedef', 'memo1', 'memo2', 'doc', 'maxform1';
18397:    AllowedChars: array[0..5] of string = ('(',')','[',']','{','}', 't,t1,t2,t3: boolean;
18398:
18399:    //-----
18400:    //*****End of mX4 Public Tools API *****
18401:    //-----
18402:
18403:    Amount of Functions: 11963
18404:    Amount of Procedures: 7474
18405:    Amount of Constructors: 1231
18406:    Totals of Calls: 20668
18407:    SHA1: Win 3.9.9.92 61ECB8AF3FA0C72B6AA6DE8DA4A65E1C30F0EE6C
18408:
18409:    *****
18410:    Doc Short Manual with 50 Tips!
18411:    *****
18412:    - Install: just save your maxboxdef.ini before and then extract the zip file!
18413:    - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
18414:    - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
18415:    - Menu: With <Ctrl><F3> you can search for code on examples
18416:    - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
18417:    - Menu: Set Interface Navigator in menu /View/Intf Navigator
18418:    - Menu: Switch or toggle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
18419:
18420:    - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
18421:    - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
18422:    - Context Menu: You can printout your scripts as a pdf-file or html-export

```

```

18423: - Context: You do have a context menu with the right mouse click
18424:
18425: - Menu: With the UseCase Editor you can convert graphic formats too.
18426: - Menu: On menu Options you find Addons as compiled scripts
18427: - IDE: You don't need a mouse to handle maxbox, use shortcuts
18428: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
18429: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
18430: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
18431: or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
18432:
18433: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
18434: - Editor: After the end, you can write or copy notes or descriptions concerning the app or code
18435: - Code: If you code a loop till key-pressed use function: isKeyPressed;
18436: - Code: Macro set the macros #name, #date, #host, #path, #file, #head #sign, Tutorial maxbox_starter25.pdf
18437: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
18438: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
18439: to delete and Click and mark to drag a bookmark
18440: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
18441: - IDE: A file info with system and script information you find in menu Program/Information
18442: - IDE: After change the config file in help you can update changes in menu Help/Config Update
18443: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
18444: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
18445: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
18446: - Editor: Set Bookmarks to check your work in app or code
18447: - Editor: With <Ctrl H> you set {$Active Line Color}, and F11 you get Word Count Statistic on Output too
18448: - Editor: With {/TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
18449: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
18450:
18451: - IDE with menu /Options/ADO SQL Workbench you can manage your Database
18452: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
18453: - Menu: Set Interface Navigagator also with toggle <Ctrl L> or /View/Intf Navigator
18454: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
18455: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
18456: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
18457: - IDE set bookmarks with <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
18458: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
18459:
18460: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
18461: - Add on when no browser is available start /Options/Add ons/Easy Browser
18462: - Add on SOAP Tester with SOP POST File
18463: - Add on IP Protocol Sniffer with List View
18464: - Add on OpenGL mX Robot Demo for android
18465:
18466: - Menu: Help/Tools as a Tool Section with DOS Opener
18467: - Menu Editor: export the code as RTF File
18468: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
18469: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
18470: - Context: Auto Detect of Syntax depending on file extension
18471: - Code: some Windows API function start with w in the name like wGetAtomName();
18472: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
18473: - IDE File Check with menu ../View/File Changes/...
18474:
18475: - using DLL example in maxbox: //function: {*****}
18476: Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
18477: cb: DWORD): BOOL; //stdcall;
18478: External 'GetProcessMemoryInfo@psapi.dll stdcall';
18479:
18480: Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
18481: External 'OpenProcess@kernel32.dll stdcall';
18482:
18483: PCT Precompile Technology , mX4 ScriptStudio
18484: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
18485: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
18486: emax layers: system-package-component-unit-class-function-block
18487: new keywords def ref using maxCalcF
18488: UML: use case act class state seq pac comp dep - lib lab
18489: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
18490: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
18491: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
18492: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
18493: https://unibe-ch.academia.edu/MaxKleiner
18494: www.slideshare.net/maxkleiner1
18495: http://www.scribd.com/max_kleiner
18496:
18497:
18498: *****
18499: unit List asm internal end
18500: *****
18501: 01 unit RRegister_StrUtils_Routines(exec); //Delphi
18502: 02 unit SRegister_IdStrings //Indy Sockets
18503: 03 unit RRegister_niSTRING_Routines(Exec); //from RegEx
18504: 04 unit uPSI_fMain Functions; //maxbox Open Tools API
18505: 05 unit IFSI_WinFormlpuzzle; //maxbox
18506: 06 unit RRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
18507: 07 unit RegisterDateTimeLibrary_R(exec); //Delphi
18508: 08 unit RRegister_MathMax_Routines(exec); //Jedi & Delphi
18509: 09 unit RRegister_IdGlobal_Routines(exec); //Indy Sockets
18510: 10 unit RRegister_SysUtils_Routines(Exec); //Delphi
18511: 11 unit uPSI_IdTCPConnection; //Indy some functions

```

```

18512: 12 unit uPSCompiler.pas; //PS kernel functions
18513: 13 unit uPSI_DBCommon; //DB Common_Routines and Types
18514: 14 unit uPSI_Printers.pas //Delphi VCL
18515: 15 unit uPSI_MPlayer.pas //Delphi VCL
18516: 16 unit uPSC_comobj; //COM Functions
18517: 17 unit uPSI_Clipbrd; //Delphi VCL
18518: 18 unit Filectrl in IFSI_SysUtils_max; //VCL Runtime
18519: 19 unit uPSI_SqlExpr; //DBX3
18520: 20 unit uPSI_ADOdb; //ADOdb
18521: 21 unit uPSI_StrHlpr; //String Helper Routines
18522: 22 unit uPSI_DateUtils; //Expansion to DateTimeLib
18523: 23 unit uPSI_FileUtils; //Expansion to Sys/File Utils
18524: 24 unit JUtils / gsUtils; //Jedi / Metabase
18525: 25 unit JvFunctions_max; //Jedi Functions
18526: 26 unit HTTPParser; //Delphi VCL
18527: 27 unit HTTPUtil; //Delphi VCL
18528: 28 unit uPSI_XMLUtil; //Delphi VCL
18529: 29 unit uPSI_SOAPHTTIClient; //Delphi VCL SOAP Webservice V3.5
18530: 30 unit uPSI_Contrns; //Delphi RTL Container of Classes
18531: 31 unit uPSI_MaskUtils; //RTL Edit and Mask functions
18532: 32 unit uPSI_MyBigInt; //big integer class with Math
18533: 33 unit uPSI_ConvUtils; //Delphi VCL Conversions engine
18534: 34 unit Types_Variants; //Delphi\Win32\rtl\sys
18535: 35 unit uPSI_IdHashSHA1; //Indy Crypto Lib
18536: 36 unit uPSI_IdHashMessageDigest //Indy Crypto;
18537: 37 unit uPSI_IdASN1Util; //Indy ASN1Utility Routines;
18538: 38 unit uPSI_IdLogFile; //Indy Logger from LogBase
18539: 39 unit uPSI_IdIcmpClient; //Indy Ping ICMP
18540: 40 unit uPSI_IdHashMessageDigest_max //Indy Crypto &OpenSSL;
18541: 41 unit uPSI_FileCtrl; //Delphi RTL
18542: 42 unit uPSI_Outline; //Delphi VCL
18543: 43 unit uPSI_ScktComp; //Delphi RTL
18544: 44 unit uPSI_Calendar; //Delphi VCL
18545: 45 unit uPSI_VListView //VListView;
18546: 46 unit uPSI_DBGGrids; //Delphi VCL
18547: 47 unit uPSI_DBCtrls; //Delphi VCL
18548: 48 unit ide_debugoutput; //maxbox
18549: 49 unit uPSI_ComCtrls; //Delphi VCL
18550: 50 unit uPSC_stdctrls+; //Delphi VCL
18551: 51 unit uPSI_Dialogs; //Delphi VCL
18552: 52 unit uPSI_StdConvs; //Delphi RTL
18553: 53 unit uPSI_DBClient; //Delphi RTL
18554: 54 unit uPSI_DBPlatform; //Delphi RTL
18555: 55 unit uPSI_Provider; //Delphi RTL
18556: 56 unit uPSI_FMTBcd; //Delphi RTL
18557: 57 unit uPSI_DBCGrids; //Delphi VCL
18558: 58 unit uPSI_CDSUtil; //MIDAS
18559: 59 unit uPSI_VarHlpr; //Delphi RTL
18560: 60 unit uPSI_ExtDlgs; //Delphi VCL
18561: 61 unit sdpStopwatch; //maxbox
18562: 62 unit uPSI_JclStatistics; //JCL
18563: 63 unit uPSI_JclLogic; //JCL
18564: 64 unit uPSI_JclMiscel; //JCL
18565: 65 unit uPSI_JclMath_max; //JCL RTL
18566: 66 unit uPSI_uTPLb_StreamUtils; //LockBox 3
18567: 67 unit uPSI_MathUtils; //BCB
18568: 68 unit uPSI_JclMultimedia; //JCL
18569: 69 unit uPSI_WideStrUtils; //Delphi API/RTL
18570: 70 unit uPSI_GraphUtil; //Delphi RTL
18571: 71 unit uPSI_TypeTrans; //Delphi RTL
18572: 72 unit uPSI_HTTPApp; //Delphi VCL
18573: 73 unit uPSI_DBWeb; //Delphi VCL
18574: 74 unit uPSI_DBBdeWeb; //Delphi VCL
18575: 75 unit uPSI_DBXpressWeb; //Delphi VCL
18576: 76 unit uPSI_ShadowWnd; //Delphi VCL
18577: 77 unit uPSI_ToolWin; //Delphi VCL
18578: 78 unit uPSI_Tabs; //Delphi VCL
18579: 79 unit uPSI_JclGraphUtils; //JCL
18580: 80 unit uPSI_JclCounter; //JCL
18581: 81 unit uPSI_JclSysInfo; //JCL
18582: 82 unit uPSI_JclSecurity; //JCL
18583: 83 unit uPSI_JclFileUtils; //JCL
18584: 84 unit uPSI_IdUserAccounts; //Indy
18585: 85 unit uPSI_IdAuthentication; //Indy
18586: 86 unit uPSI_uTPLb_AES; //LockBox 3
18587: 87 unit uPSI_IdHashSHA1; //LockBox 3
18588: 88 unit uTPLb_BlockCipher; //LockBox 3
18589: 89 unit uPSI_ValEdit.pas; //Delphi VCL
18590: 90 unit uPSI_JvVCLUtils; //JCL
18591: 91 unit uPSI_JvDBUtil; //JCL
18592: 92 unit uPSI_JvDBUtils; //JCL
18593: 93 unit uPSI_JvAppUtils; //JCL
18594: 94 unit uPSI_JvCtrlUtils; //JCL
18595: 95 unit uPSI_JvFormToHtml; //JCL
18596: 96 unit uPSI_JvParsing; //JCL
18597: 97 unit uPSI_SerDlgs; //Toolbox
18598: 98 unit uPSI_Serial; //Toolbox
18599: 99 unit uPSI_JvComponent; //JCL
18600: 100 unit uPSI_JvCalc; //JCL

```

```

18601: 101 unit uPSI_JvBdeUtils; //JCL
18602: 102 unit uPSI_JvDateUtil; //JCL
18603: 103 unit uPSI_JvGenetic; //JCL
18604: 104 unit uPSI_JclBase; //JCL
18605: 105 unit uPSI_JvUtils; //JCL
18606: 106 unit uPSI_JvStrUtil; //JCL
18607: 107 unit uPSI_JvStrUtils; //JCL
18608: 108 unit uPSI_JvFileUtil; //JCL
18609: 109 unit uPSI_JvMemoryInfos; //JCL
18610: 110 unit uPSI_JvComputerInfo; //JCL
18611: 111 unit uPSI_JvgCommClasses; //JCL
18612: 112 unit uPSI_JvgLogics; //JCL
18613: 113 unit uPSI_JvLED; //JCL
18614: 114 unit uPSI_JvTurtle; //JCL
18615: 115 unit uPSI_SortThds; unit uPSI_ThSort; //maXbox
18616: 116 unit uPSI_JvgUtils; //JCL
18617: 117 unit uPSI_JvExprParser; //JCL
18618: 118 unit uPSI_HexDump; //Borland
18619: 119 unit uPSI_DBLogDlg; //VCL
18620: 120 unit uPSI_SqlTimSt; //RTL
18621: 121 unit uPSI_JvHtmlParser; //JCL
18622: 122 unit uPSI_JvgXMLSerializer; //JCL
18623: 123 unit uPSI_JvJCLUtils; //JCL
18624: 124 unit uPSI_JvStrings; //JCL
18625: 125 unit uPSI_uTPLb_IntegerUtils; //TurboPower
18626: 126 unit uPSI_uTPLb_HugeCardinal; //TurboPower
18627: 127 unit uPSI_uTPLb_HugeCardinalUtils; //TurboPower
18628: 128 unit uPSI_SynRegExpr; //SynEdit
18629: 129 unit uPSI_StUtils; //SysTools4
18630: 130 unit uPSI_StToHTML; //SysTools4
18631: 131 unit uPSI_StStrms; //SysTools4
18632: 132 unit uPSI_StFIN; //SysTools4
18633: 133 unit uPSI_StAstroP; //SysTools4
18634: 134 unit uPSI_StStat; //SysTools4
18635: 135 unit uPSI_StNetCon; //SysTools4
18636: 136 unit uPSI_StDecMth; //SysTools4
18637: 137 unit uPSI_StOStr; //SysTools4
18638: 138 unit uPSI_StPtrns; //SysTools4
18639: 139 unit uPSI_StNetMessage; //SysTools4
18640: 140 unit uPSI_StMath; //SysTools4
18641: 141 unit uPSI_StExpEng; //SysTools4
18642: 142 unit uPSI_StCRC; //SysTools4
18643: 143 unit uPSI_StExport; //SysTools4
18644: 144 unit uPSI_StExpLog; //SysTools4
18645: 145 unit uPSI_ActnList; //Delphi VCL
18646: 146 unit uPSI_jpeg; //Borland
18647: 147 unit uPSI_StRandom; //SysTools4
18648: 148 unit uPSI_StDict; //SysTools4
18649: 149 unit uPSI_StBCD; //SysTools4
18650: 150 unit uPSI_StTxtDat; //SysTools4
18651: 151 unit uPSI_StRegEx; //SysTools4
18652: 152 unit uPSI_IMouse; //VCL
18653: 153 unit uPSI_SyncObjs; //VCL
18654: 154 unit uPSI_AsyncCalls; //Hausladen
18655: 155 unit uPSI_ParallelJobs; //Saraiva
18656: 156 unit uPSI_Variants; //VCL
18657: 157 unit uPSI_VarCmplx; //VCL Wolfram
18658: 158 unit uPSI_DTDSchema; //VCL
18659: 159 unit uPSI_ShLwApi; //Brakel
18660: 160 unit uPSI_IBUtils; //VCL
18661: 161 unit uPSI_CheckLst; //VCL
18662: 162 unit uPSI_JvSimpleXml; //JCL
18663: 163 unit uPSI_JclSimpleXml; //JCL
18664: 164 unit uPSI_JvXmlDatabase; //JCL
18665: 165 unit uPSI_JvMaxPixel; //JCL
18666: 166 unit uPSI_JvItemsSearchs; //JCL
18667: 167 unit uPSI_StExpEng2; //SysTools4
18668: 168 unit uPSI_StGenLog; //SysTools4
18669: 169 unit uPSI_JvLogFile; //Jcl
18670: 170 unit uPSI_CPort; //ComPort Lib v4.11
18671: 171 unit uPSI_CPortCtl; //ComPort
18672: 172 unit uPSI_CPortEsc; //ComPort
18673: 173 unit BarCodeScanner; //ComPort
18674: 174 unit uPSI_JvGraph; //JCL
18675: 175 unit uPSI_JvComCtrls; //JCL
18676: 176 unit uPSI_GUITesting; //D Unit
18677: 177 unit uPSI_JvFindFiles; //JCL
18678: 178 unit uPSI_StSystem; //SysTools4
18679: 179 unit uPSI_JvKeyboardStates; //JCL
18680: 180 unit uPSI_JvMail; //JCL
18681: 181 unit uPSI_JclConsole; //JCL
18682: 182 unit uPSI_JclLANMan; //JCL
18683: 183 unit uPSI_IdCustomHTTPServer; //Indy
18684: 184 unit IdHTTPServer; //Indy
18685: 185 unit uPSI_IdTCPServer; //Indy
18686: 186 unit uPSI_IdSocketHandle; //Indy
18687: 187 unit uPSI_IdIOHandlerSocket; //Indy
18688: 188 unit IdIOHandler; //Indy
18689: 189 unit uPSI_cutils; //Bloodshed

```



```

18690: 190 unit uPSI_BoldUtils; //boldsoft
18691: 191 unit uPSI_IdSimpleServer; //Indy
18692: 192 unit uPSI_IdSSLOpenSSL; //Indy
18693: 193 unit uPSI_IdMultipartFormData; //Indy
18694: 194 unit uPSI_SynURIOpener; //SynEdit
18695: 195 unit uPSI_PerlRegEx; //PCRE
18696: 196 unit uPSI_IdHeaderList; //Indy
18697: 197 unit uPSI_StFirst; //SysTools4
18698: 198 unit uPSI_JvCtrls; //JCL
18699: 199 unit uPSI_IdTrivialFTPBase; //Indy
18700: 200 unit uPSI_IdTrivialFTP; //Indy
18701: 201 unit uPSI_IdUDPBase; //Indy
18702: 202 unit uPSI_IdUDPCClient; //Indy
18703: 203 unit uPSI_utypes; //for DMath.DLL
18704: 204 unit uPSI_ShellAPI; //Borland
18705: 205 unit uPSI_IdRemoteCMDClient; //Indy
18706: 206 unit uPSI_IdRemoteCMDServer; //Indy
18707: 207 unit IdRexecServer; //Indy
18708: 208 unit IdRexec; (unit uPSI_IdRexec;) //Indy
18709: 209 unit IdUDPServer; //Indy
18710: 210 unit IdTimeUDPServer; //Indy
18711: 211 unit IdTimeServer; //Indy
18712: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
18713: 213 unit uPSI_IdIPWatch; //Indy
18714: 214 unit uPSI_IdIrcServer; //Indy
18715: 215 unit uPSI_IdMessageCollection; //Indy
18716: 216 unit uPSI_cPEM; //Fundamentals 4
18717: 217 unit uPSI_cFundamentUtils; //Fundamentals 4
18718: 218 unit uPSI_uwinplot; //DMath
18719: 219 unit uPSI_xrtl_util_CPUUtils; //ExtendedRTL
18720: 220 unit uPSI_GR32_System; //Graphics32
18721: 221 unit uPSI_cFileUtils; //Fundamentals 4
18722: 222 unit uPSI_cDateTime; (timemachine) //Fundamentals 4
18723: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
18724: 224 unit uPSI_cRandom; //Fundamentals 4
18725: 225 unit uPSI_ueval; //DMath
18726: 226 unit uPSI_xrtl_net_URIUtils; //ExtendedRTL
18727: 227 unit xrtl_net_URIUtils; //ExtendedRTL
18728: 228 unit uPSI_ufft; (FFT) //DMath
18729: 229 unit uPSI_DBXChannel; //Delphi
18730: 230 unit uPSI_DBXIndyChannel; //Delphi Indy
18731: 231 unit uPSI_xrtl_util_COMCat; //ExtendedRTL
18732: 232 unit uPSI_xrtl_util_StrUtils; //ExtendedRTL
18733: 233 unit uPSI_xrtl_util_VariantUtils; //ExtendedRTL
18734: 234 unit uPSI_xrtl_util_FileUtils; //ExtendedRTL
18735: 235 unit xrtl_util_Compat; //ExtendedRTL
18736: 236 unit uPSI_OleAuto; //Borland
18737: 237 unit uPSI_xrtl_util_COMUtils; //ExtendedRTL
18738: 238 unit uPSI_CmAdmCtl; //Borland
18739: 239 unit uPSI_ValEdit2; //VCL
18740: 240 unit uPSI_GR32; //Graphics32
18741: 241 unit uPSI_GR32_Image; //Graphics32
18742: 242 unit uPSI_xrtl_util_TimeUtils; //ExtendedRTL
18743: 243 unit uPSI_xrtl_util_TimeZone; //ExtendedRTL
18744: 244 unit uPSI_xrtl_util_TimeStamp; //ExtendedRTL
18745: 245 unit uPSI_xrtl_util_Map; //ExtendedRTL
18746: 246 unit uPSI_xrtl_util_Set; //ExtendedRTL
18747: 247 unit uPSI_CPortMonitor; //ComPort
18748: 248 unit uPSI_StIniStm; //SysTools4
18749: 249 unit uPSI_GR32_ExtImage; //Graphics32
18750: 250 unit uPSI_GR32_OrdinalMaps; //Graphics32
18751: 251 unit uPSI_GR32_Rasterizers; //Graphics32
18752: 252 unit uPSI_xrtl_util_Exception; //ExtendedRTL
18753: 253 unit uPSI_xrtl_util_Value; //ExtendedRTL
18754: 254 unit uPSI_xrtl_util_Compare; //ExtendedRTL
18755: 255 unit uPSI_FlatSB; //VCL
18756: 256 unit uPSI_JvAnalogClock; //JCL
18757: 257 unit uPSI_JvAlarms; //JCL
18758: 258 unit uPSI_JvSQLS; //JCL
18759: 259 unit uPSI_JvDBSecur; //JCL
18760: 260 unit uPSI_JvDBQBE; //JCL
18761: 261 unit uPSI_JvStarfield; //JCL
18762: 262 unit uPSI_JvCLMiscal; //JCL
18763: 263 unit uPSI_JvProfiler32; //JCL
18764: 264 unit uPSI_JvDirectories; //JCL
18765: 265 unit uPSI_JclSchedule; //JCL
18766: 266 unit uPSI_JclSvcCtrl; //JCL
18767: 267 unit uPSI_JvSoundControl; //JCL
18768: 268 unit uPSI_JvBDESQLScript; //JCL
18769: 269 unit uPSI_JvgDigits; //JCL>
18770: 270 unit uPSI_ImgList; //TCustomImageList
18771: 271 unit uPSI_JclMIDI; //JCL>
18772: 272 unit uPSI_JclWinMidi; //JCL>
18773: 273 unit uPSI_JclNTFS; //JCL>
18774: 274 unit uPSI_JclAppInst; //JCL>
18775: 275 unit uPSI_JvRle; //JCL>
18776: 276 unit uPSI_JvRas32; //JCL>
18777: 277 unit uPSI_JvImageDrawThread; //JCL>
18778: 278 unit uPSI_JvImageWindow; //JCL>

```

```

18779: 279 unit uPSI_JvTransparentForm; //JCL>
18780: 280 unit uPSI_JvWinDialogs; //JCL>
18781: 281 unit uPSI_JvSimLogic; //JCL>
18782: 282 unit uPSI_JvSimIndicator; //JCL>
18783: 283 unit uPSI_JvSimPID; //JCL>
18784: 284 unit uPSI_JvSimPIDLinker; //JCL>
18785: 285 unit uPSI_IdRFCReply; //Indy
18786: 286 unit uPSI_IdIdent; //Indy
18787: 287 unit uPSI_IdIdentServer; //Indy
18788: 288 unit uPSI_JvPatchFile; //JCL
18789: 289 unit uPSI_StNetPfm; //SysTools4
18790: 290 unit uPSI_StNet; //SysTools4
18791: 291 unit uPSI_JclPeImage; //JCL
18792: 292 unit uPSI_JclPrint; //JCL
18793: 293 unit uPSI_JclMime; //JCL
18794: 294 unit uPSI_JvRichEdit; //JCL
18795: 295 unit uPSI_JvDBRichEd; //JCL
18796: 296 unit uPSI_JvDice; //JCL
18797: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
18798: 298 unit uPSI_JvDirFrm; //JCL
18799: 299 unit uPSI_JvDualList; //JCL
18800: 300 unit uPSI_JvSwitch; ///JCL
18801: 301 unit uPSI_JvTimerLst; ///JCL
18802: 302 unit uPSI_JvMemTable; //JCL
18803: 303 unit uPSI_JvObjStr; //JCL
18804: 304 unit uPSI_StLArr; //SysTools4
18805: 305 unit uPSI_StWmDCpy; //SysTools4
18806: 306 unit uPSI_StText; //SysTools4
18807: 307 unit uPSI_StNTLog; //SysTools4
18808: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL
18809: 309 unit uPSI_JvImagPrvw; //JCL
18810: 310 unit uPSI_JvFormPatch; //JCL
18811: 311 unit uPSI_JvPicClip; //JCL
18812: 312 unit uPSI_JvDataConv; //JCL
18813: 313 unit uPSI_JvCpuUsage; //JCL
18814: 314 unit uPSI_JclUnitConv_mX2; //JCL
18815: 315 unit JvDualListForm; //JCL
18816: 316 unit uPSI_JvCpuUsage2; //JCL
18817: 317 unit uPSI_JvParserForm; //JCL
18818: 318 unit uPSI_JvJanTreeView; //JCL
18819: 319 unit uPSI_JvTransLED; //JCL
18820: 320 unit uPSI_JvPlaylist; //JCL
18821: 321 unit uPSI_JvFormAutoSize; //JCL
18822: 322 unit uPSI_JvYearGridEditForm; //JCL
18823: 323 unit uPSI_JvMarkupCommon; //JCL
18824: 324 unit uPSI_JvChart; //JCL
18825: 325 unit uPSI_JvXPCore; //JCL
18826: 326 unit uPSI_JvXPCoreUtils; //JCL
18827: 327 unit uPSI_StatsClasses; //mX4
18828: 328 unit uPSI_ExtCtrls2; //VCL
18829: 329 unit uPSI_JvUrlGrabbers; //JCL
18830: 330 unit uPSI_JvXmlTree; //JCL
18831: 331 unit uPSI_JvWavePlayer; //JCL
18832: 332 unit uPSI_JvUnicodeCanvas; //JCL
18833: 333 unit uPSI_JvTFUtils; //JCL
18834: 334 unit uPSI_IdServerIOHandler; //Indy
18835: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
18836: 336 unit uPSI_IdMessageCoder; //Indy
18837: 337 unit uPSI_IdMessageCoderMIME; //Indy
18838: 338 unit uPSI_IdMIMETypes; //Indy
18839: 339 unit uPSI_JvConverter; //JCL
18840: 340 unit uPSI_JvCsvParse; //JCL
18841: 341 unit uPSI_umath; unit uPSI_ugamma; //DMath
18842: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
18843: 343 unit uPSI_JvDBGridExport; //JCL
18844: 344 unit uPSI_JvgExport; //JCL
18845: 345 unit uPSI_JvSerialMaker; //JCL
18846: 346 unit uPSI_JvWin32; //JCL
18847: 347 unit uPSI_JvPaintFX; //JCL
18848: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
18849: 349 unit uPSI_JvValidators; (preview) //JCL
18850: 350 unit uPSI_JvNTEventLog; //JCL
18851: 351 unit uPSI_ShellZipTool; //mX4
18852: 352 unit uPSI_JvJoystick; //JCL
18853: 353 unit uPSI_JvMailSlots; //JCL
18854: 354 unit uPSI_JclComplex; //JCL
18855: 355 unit uPSI_SynPdf; //Synopsis
18856: 356 unit uPSI_Registry; //VCL
18857: 357 unit uPSI_TlHelp32; //VCL
18858: 358 unit uPSI_JclRegistry; //JCL
18859: 359 unit uPSI_JvAirBrush; //JCL
18860: 360 unit uPSI_mORMotReport; //Synopsis
18861: 361 unit uPSI_JclLocales; //JCL
18862: 362 unit uPSI_SynEdit; //SynEdit
18863: 363 unit uPSI_SynEditTypes; //SynEdit
18864: 364 unit uPSI_SynMacroRecorder; //SynEdit
18865: 365 unit uPSI_LongIntList; //SynEdit
18866: 366 unit uPSI_devcutils; //DevC
18867: 367 unit uPSI_SynEditMiscClasses; //SynEdit

```

```

18868: 368 unit uPSI_SynEditRegexSearch; //SynEdit
18869: 369 unit uPSI_SynEditHighlighter; //SynEdit
18870: 370 unit uPSI_SynHighlighterPas; //SynEdit
18871: 371 unit uPSI_JvSearchFiles; //JCL
18872: 372 unit uPSI_SynHighlighterAny; //Lazarus
18873: 373 unit uPSI_SynEditKeyCmds; //SynEdit
18874: 374 unit uPSI_SynEditMiscProcs; //SynEdit
18875: 375 unit uPSI_SynEditKbdHandler //SynEdit
18876: 376 unit uPSI_JvAppInst; //JCL
18877: 377 unit uPSI_JvAppEvent; //JCL
18878: 378 unit uPSI_JvAppCommand; //JCL
18879: 379 unit uPSI_JvAnimTitle; //JCL
18880: 380 unit uPSI_JvAnimatedImage; //JCL
18881: 381 unit uPSI_SynEditExport; //SynEdit
18882: 382 unit uPSI_SynExportHTML; //SynEdit
18883: 383 unit uPSI_SynExportRTF; //SynEdit
18884: 384 unit uPSI_SynEditSearch; //SynEdit
18885: 385 unit uPSI_fMain_back //maXbox;
18886: 386 unit uPSI_JvZoom; //JCL
18887: 387 unit uPSI_PMrand; //PM
18888: 388 unit uPSI_JvSticker; //JCL
18889: 389 unit uPSI_XmlVerySimple; //mX4
18890: 390 unit uPSI_Services; //ExtPascal
18891: 391 unit uPSI_ExtPascalUtils; //ExtPascal
18892: 392 unit uPSI_SocketsDelphi; //ExtPascal
18893: 393 unit uPSI_StBarC; //SysTools
18894: 394 unit uPSI_StDbBarC; //SysTools
18895: 395 unit uPSI_StBarPN; //SysTools
18896: 396 unit uPSI_StDbPNBC; //SysTools
18897: 397 unit uPSI_StDb2DBC; //SysTools
18898: 398 unit uPSI_StMoney; //SysTools
18899: 399 unit uPSI_JvForth; //JCL
18900: 400 unit uPSI_RestRequest; //mX4
18901: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
18902: 402 unit uPSI_JvXmlDatabase; //update //JCL
18903: 403 unit uPSI_StAstro; //SysTools
18904: 404 unit uPSI_StSort; //SysTools
18905: 405 unit uPSI_StDate; //SysTools
18906: 406 unit uPSI_StDateSt; //SysTools
18907: 407 unit uPSI_StBase; //SysTools
18908: 408 unit uPSI_StVInfo; //SysTools
18909: 409 unit uPSI_JvBrowseFolder; //JCL
18910: 410 unit uPSI_JvBoxProcs; //JCL
18911: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
18912: 412 unit uPSI_usimann; (unit ugenalg;) //DMath
18913: 413 unit uPSI_JvHighlighter; //JCL
18914: 414 unit uPSI_Diff; //mX4
18915: 415 unit uPSI_SpringWinAPI; //DSpring
18916: 416 unit uPSI_StBits; //SysTools
18917: 417 unit uPSI_TomDBQueue; //mX4
18918: 418 unit uPSI_MultilangTranslator; //mX4
18919: 419 unit uPSI_HyperLabel; //mX4
18920: 420 unit uPSI_Starter; //mX4
18921: 421 unit uPSI_FileAssocs; //devC
18922: 422 unit uPSI_devFileMonitorX; //devC
18923: 423 unit uPSI_devrun; //devC
18924: 424 unit uPSI_devExec; //devC
18925: 425 unit uPSI_oysUtils; //devC
18926: 426 unit uPSI_DosCommand; //devC
18927: 427 unit uPSI_CppTokenizer; //devC
18928: 428 unit uPSI_JvHLLParser; //devC
18929: 429 unit uPSI_JclMapi; //JCL
18930: 430 unit uPSI_JclShell; //JCL
18931: 431 unit uPSI_JclCOM; //JCL
18932: 432 unit uPSI_GR32_Math; //Graphics32
18933: 433 unit uPSI_GR32_LowLevel; //Graphics32
18934: 434 unit uPSI_SimpleHl; //mX4
18935: 435 unit uPSI_GR32_Filters; //Graphics32
18936: 436 unit uPSI_GR32_VectorMaps; //Graphics32
18937: 437 unit uPSI_cXMLFunctions; //Fundamentals 4
18938: 438 unit uPSI_JvTimer; //JCL
18939: 439 unit uPSI_cHTTPUTils; //Fundamentals 4
18940: 440 unit uPSI_cTLSTils; //Fundamentals 4
18941: 441 unit uPSI_JclGraphics; //JCL
18942: 442 unit uPSI_JclSynch; //JCL
18943: 443 unit uPSI_IdTelnet; //Indy
18944: 444 unit uPSI_IdTelnetServer; //Indy
18945: 445 unit uPSI_IdEcho; //Indy
18946: 446 unit uPSI_IdEchoServer; //Indy
18947: 447 unit uPSI_IdEchoUDP; //Indy
18948: 448 unit uPSI_IdEchoUDPServer; //Indy
18949: 449 unit uPSI_IdSocks; //Indy
18950: 450 unit uPSI_IdAntiFreezeBase; //Indy
18951: 451 unit uPSI_IdHostnameServer; //Indy
18952: 452 unit uPSI_IdTunnelCommon; //Indy
18953: 453 unit uPSI_IdTunnelMaster; //Indy
18954: 454 unit uPSI_IdTunnelSlave; //Indy
18955: 455 unit uPSI_IdRSH; //Indy
18956: 456 unit uPSI_IdRSHServer; //Indy

```

```

18957: 457 unit uPSI_Spring_Cryptography_Utils; //Spring4Delphi
18958: 458 unit uPSI_MapReader, //devC
18959: 459 unit uPSI_LibTar, //devC
18960: 460 unit uPSI_IdStack; //Indy
18961: 461 unit uPSI_IdBlockCipherIntercept; //Indy
18962: 462 unit uPSI_IdChargenServer; //Indy
18963: 463 unit uPSI_IdFTPServer; //Indy
18964: 464 unit uPSI_IdException; //Indy
18965: 465 unit uPSI_utexplot; //DMath
18966: 466 unit uPSI_uwinstr; //DMath
18967: 467 unit uPSI_VarRecUtils; //devC
18968: 468 unit uPSI_JvStringListToHtml, //JCL
18969: 469 unit uPSI_JvStringHolder, //JCL
18970: 470 unit uPSI_IdCoder; //Indy
18971: 471 unit uPSI_SynHighlighterDfm; //Synedit
18972: 472 unit uHighlighterProcs; in 471 //Synedit
18973: 473 unit uPSI_LazFileUtils, //LCL
18974: 474 unit uPSI_IDECmdLine; //LCL
18975: 475 unit uPSI_lazMasks; //LCL
18976: 476 unit uPSI_ip_misc; //mX4
18977: 477 unit uPSI_Barcode; //LCL
18978: 478 unit uPSI_SimpleXML; //LCL
18979: 479 unit uPSI_JclIniFiles; //JCL
18980: 480 unit uPSI_D2XXUnit; {$X-} //FTDI
18981: 481 unit uPSI_JclDateTime; //JCL
18982: 482 unit uPSI_JclEDI; //JCL
18983: 483 unit uPSI_JclMiscel2; //JCL
18984: 484 unit uPSI_JclValidation; //JCL
18985: 485 unit uPSI_JclAnsiStrings; {-PString} //JCL
18986: 486 unit uPSI_SynEditMiscProcs2; //Synedit
18987: 487 unit uPSI_JclStreams; //JCL
18988: 488 unit uPSI_QRCode; //mX4
18989: 489 unit uPSI_BlockSocket; //ExtPascal
18990: 490 unit uPSI_Masks,Utills //VCL
18991: 491 unit uPSI_synauttil; //Synapse!
18992: 492 unit uPSI_JclMath_Class; //JCL RTL
18993: 493 unit ugamdist; //Gamma function //DMath
18994: 494 unit ubeta, ucorrel; //IBeta //DMath
18995: 495 unit uPSI_SRMgr; //mX4
18996: 496 unit uPSI_HotLog; //mX4
18997: 497 unit uPSI_DebugBox; //mX4
18998: 498 unit uPSI_ustrings; //DMath
18999: 499 unit uPSI_uregtest; //DMath
19000: 500 unit uPSI_usimplex; //DMath
19001: 501 unit uPSI_uhyper; //DMath
19002: 502 unit uPSI_IdHLL7; //Indy
19003: 503 unit uPSI_IdIPMCastBase, //Indy
19004: 504 unit uPSI_IdIPMCastServer; //Indy
19005: 505 unit uPSI_IdIPMCastClient; //Indy
19006: 506 unit uPSI_unlfit; //nlregression //DMath
19007: 507 unit uPSI_IdRawHeaders; //Indy
19008: 508 unit uPSI_IdRawClient; //Indy
19009: 509 unit uPSI_IdRawFunctions; //Indy
19010: 510 unit uPSI_IdTCPStream; //Indy
19011: 511 unit uPSI_IdSNPP; //Indy
19012: 512 unit uPSI_St2DBarC; //SysTools
19013: 513 unit uPSI_ImageWin; //FTL //VCL
19014: 514 unit uPSI_CustomDrawTreeView; //FTL //VCL
19015: 515 unit uPSI_GraphWin; //FTL //VCL
19016: 516 unit uPSI_actionMain; //FTL //VCL
19017: 517 unit uPSI_StSpawn; //SysTools
19018: 518 unit uPSI_CtlPanel; //VCL
19019: 519 unit uPSI_IdLPR; //Indy
19020: 520 unit uPSI_SockRequestInterpreter; //Indy
19021: 521 unit uPSI_ulambert; //DMath
19022: 522 unit uPSI_ucholesk; //DMath
19023: 523 unit uPSI_SimpleDS; //VCL
19024: 524 unit uPSI_DBXSqlScanner; //VCL
19025: 525 unit uPSI_DBXMetadataUtil; //VCL
19026: 526 unit uPSI_Chart; //TEE
19027: 527 unit uPSI_TeeProcs; //TEE
19028: 528 unit mXBDEUtils; //mX4
19029: 529 unit uPSI_MDIEdit; //VCL
19030: 530 unit uPSI_CopyPrsr; //VCL
19031: 531 unit uPSI_SockApp; //VCL
19032: 532 unit uPSI_AppEvnts; //VCL
19033: 533 unit uPSI_ExtActns; //VCL
19034: 534 unit uPSI_TeEngine; //TEE
19035: 535 unit uPSI_CoolMain; //browser //VCL
19036: 536 unit uPSI_StCRC; //SysTools
19037: 537 unit uPSI_StDecMth2; //SysTools
19038: 538 unit uPSI_frmExportMain; //Synedit
19039: 539 unit uPSI_SynDBEdit; //Synedit
19040: 540 unit uPSI_SynEditWildcardSearch; //Synedit
19041: 541 unit uPSI_BoldComUtils; //BOLD
19042: 542 unit uPSI_BoldIsoDateTime; //BOLD
19043: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
19044: 544 unit uPSI_BoldXMLRequests; //BOLD
19045: 545 unit uPSI_BoldStringList; //BOLD

```



```

19046: 546 unit uPSI_BoldFileHandler; //BOLD
19047: 547 unit uPSI_BoldContainers; //BOLD
19048: 548 unit uPSI_BoldQueryUserDlg; //BOLD
19049: 549 unit uPSI_BoldWinINet; //BOLD
19050: 550 unit uPSI_BoldQueue; //BOLD
19051: 551 unit uPSI_JvPcx; //JCL
19052: 552 unit uPSI_IdWhois; //Indy
19053: 553 unit uPSI_IdWhoIsServer; //Indy
19054: 554 unit uPSI_IdGopher; //Indy
19055: 555 unit uPSI_IdDateTimeStamp; //Indy
19056: 556 unit uPSI_IdDayTimeServer; //Indy
19057: 557 unit uPSI_IdDayTimeUDP; //Indy
19058: 558 unit uPSI_IdDayTimeUDPServer; //Indy
19059: 559 unit uPSI_IdDICTServer; //Indy
19060: 560 unit uPSI_IdDiscardServer; //Indy
19061: 561 unit uPSI_IdDiscardUDPServer; //Indy
19062: 562 unit uPSI_IdMappedFTP; //Indy
19063: 563 unit uPSI_IdMappedPortTCP; //Indy
19064: 564 unit uPSI_IdGopherServer; //Indy
19065: 565 unit uPSI_IdQotdServer; //Indy
19066: 566 unit uPSI_JvRgbToHtml; //JCL
19067: 567 unit uPSI_JvRemLog; //JCL
19068: 568 unit uPSI_JvSysComp; //JCL
19069: 569 unit uPSI_JvTMTL; //JCL
19070: 570 unit uPSI_JvWinampAPI; //JCL
19071: 571 unit uPSI_MSysUtils; //mX4
19072: 572 unit uPSI_ESBMaths; //ESB
19073: 573 unit uPSI_ESBMaths2; //ESB
19074: 574 unit uPSI_uLkJSON; //Lk
19075: 575 unit uPSI_ZURL; //Zeos
19076: 576 unit uPSI_ZSysUtils; //Zeos
19077: 577 unit unaUtils internals //UNA
19078: 578 unit uPSI_ZMatchPattern; //Zeos
19079: 579 unit uPSI_ZClasses; //Zeos
19080: 580 unit uPSI_ZCollections; //Zeos
19081: 581 unit uPSI_ZEncoding; //Zeos
19082: 582 unit uPSI_IdRawBase; //Indy
19083: 583 unit uPSI_IdNTLM; //Indy
19084: 584 unit uPSI_IdNNTP; //Indy
19085: 585 unit uPSI_usniffer; //PortScanForm
19086: 586 unit uPSI_IdCoderMIME; //Indy
19087: 587 unit uPSI_IdCoderUUE; //Indy
19088: 588 unit uPSI_IdCoderXXE; //Indy
19089: 589 unit uPSI_IdCoder3to4; //Indy
19090: 590 unit uPSI_IdCookie; //Indy
19091: 591 unit uPSI_IdCookieManager; //Indy
19092: 592 unit uPSI_WDosSocketUtils; //WDos
19093: 593 unit uPSI_WDosPlcUtils; //WDos
19094: 594 unit uPSI_WDosPorts; //WDos
19095: 595 unit uPSI_WDosResolvers; //WDos
19096: 596 unit uPSI_WDosTimers; //WDos
19097: 597 unit uPSI_WDosPlcs; //WDos
19098: 598 unit uPSI_WDosPneumatics; //WDos
19099: 599 unit uPSI_IdFingerServer; //Indy
19100: 600 unit uPSI_IdDNSResolver; //Indy
19101: 601 unit uPSI_IdHTTPWebBrokerBridge; //Indy
19102: 602 unit uPSI_IdIntercept; //Indy
19103: 603 unit uPSI_IdIPMCastBase; //Indy
19104: 604 unit uPSI_IdLogBase; //Indy
19105: 605 unit uPSI_IdIOHandlerStream; //Indy
19106: 606 unit uPSI_IdMappedPortUDP; //Indy
19107: 607 unit uPSI_IdQOTDUDPServer; //Indy
19108: 608 unit uPSI_IdQOTDUDP; //Indy
19109: 609 unit uPSI_IdSysLog; //Indy
19110: 610 unit uPSI_IdSysLogServer; //Indy
19111: 611 unit uPSI_IdSysLogMessage; //Indy
19112: 612 unit uPSI_IdTimeServer; //Indy
19113: 613 unit uPSI_IdTimeUDP; //Indy
19114: 614 unit uPSI_IdTimeUDPServer; //Indy
19115: 615 unit uPSI_IdUserAccounts; //Indy
19116: 616 unit uPSI_TextUtils; //mX4
19117: 617 unit uPSI_MandelbrotEngine; //mX4
19118: 618 unit uPSI_delphi_arduino_Unit1; //mX4
19119: 619 unit uPSI_DTDSchema2; //mX4
19120: 620 unit uPSI_fplotMain; //DMath
19121: 621 unit uPSI_FindFileIter; //mX4
19122: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
19123: 623 unit uPSI_PppParser; //PPP
19124: 624 unit uPSI_PppLexer; //PPP
19125: 625 unit uPSI_PCharUtils; //PPP
19126: 626 unit uPSI_uJSON; //WU
19127: 627 unit uPSI_JclStrHashMap; //JCL
19128: 628 unit uPSI_JclHookExcept; //JCL
19129: 629 unit uPSI_EncdDecd; //VCL
19130: 630 unit uPSI_SockAppReg; //VCL
19131: 631 unit uPSI_PJFileHandle; //PJ
19132: 632 unit uPSI_PJEnvVars; //PJ
19133: 633 unit uPSI_PJPipe; //PJ
19134: 634 unit uPSI_PJPipeFilters; //PJ

```

```

19135: 635 unit uPSI_PJConsoleApp; //PJ
19136: 636 unit uPSI_UConsoleAppEx; //PJ
19137: 637 unit uPSI_DbxSocketChannelNative; //VCL
19138: 638 unit uPSI_DbxDataGenerator; //VCL
19139: 639 unit uPSI_DBXClient; //VCL
19140: 640 unit uPSI_IdLogEvent; //Indy
19141: 641 unit uPSI_Reversi; //mX4
19142: 642 unit uPSI_Geometry; //mX4
19143: 643 unit uPSI_IdSMTPServer; //Indy
19144: 644 unit uPSI_Textures; //mX4
19145: 645 unit uPSI_IBX; //VCL
19146: 646 unit uPSI_IWDBCommon; //VCL
19147: 647 unit uPSI_SortGrid; //mX4
19148: 648 unit uPSI_IB; //VCL
19149: 649 unit uPSI_IBScript; //VCL
19150: 650 unit uPSI_JvCSVBaseControls; //JCL
19151: 651 unit uPSI_Jvg3DColors; //JCL
19152: 652 unit uPSI_JvHLEditor; //beat //JCL
19153: 653 unit uPSI_JvShellHook; //JCL
19154: 654 unit uPSI_DBCommon2; //VCL
19155: 655 unit uPSI_JvSHFileOperation; //JCL
19156: 656 unit uPSI_uFileExport; //mX4
19157: 657 unit uPSI_JvDialogs; //JCL
19158: 658 unit uPSI_JvDBTreeView; //JCL
19159: 659 unit uPSI_JvDBUltimGrid; //JCL
19160: 660 unit uPSI_JvDBQueryParamsForm; //JCL
19161: 661 unit uPSI_JvExControls; //JCL
19162: 662 unit uPSI_JvBDEMemTable; //JCL
19163: 663 unit uPSI_JvCommStatus; //JCL
19164: 664 unit uPSI_JvMailsSlots2; //JCL
19165: 665 unit uPSI_JvgWinMask; //JCL
19166: 666 unit uPSI_StEclipse; //SysTools
19167: 667 unit uPSI_StMime; //SysTools
19168: 668 unit uPSI_StList; //SysTools
19169: 669 unit uPSI_StMerge; //SysTools
19170: 670 unit uPSI_StStrS; //SysTools
19171: 671 unit uPSI_StTree; //SysTools
19172: 672 unit uPSI_StVArr; //SysTools
19173: 673 unit uPSI_StRegIni; //SysTools
19174: 674 unit uPSI_urkf; //DMath
19175: 675 unit uPSI_usvd; //DMath
19176: 676 unit uPSI_DepWalkUtils; //JCL
19177: 677 unit uPSI_OptionsFrm; //JCL
19178: 678 unit yuvconverts; //mX4
19179: 679 uPSI_JvPropAutoSave; //JCL
19180: 680 uPSI_AclAPI; //alcinoe
19181: 681 uPSI_AviCap; //alcinoe
19182: 682 uPSI_ALAVLBinaryTree; //alcinoe
19183: 683 uPSI_ALFcnMisc; //alcinoe
19184: 684 uPSI_ALStringList; //alcinoe
19185: 685 uPSI_ALQuickSortList; //alcinoe
19186: 686 uPSI_ALStaticText; //alcinoe
19187: 687 uPSI_ALJSONDoc; //alcinoe
19188: 688 uPSI_ALGSMComm; //alcinoe
19189: 689 uPSI_ALWindows; //alcinoe
19190: 690 uPSI_ALMultiPartFormDataParser; //alcinoe
19191: 691 uPSI_ALHttpCommon; //alcinoe
19192: 692 uPSI_ALWebSpider; //alcinoe
19193: 693 uPSI_ALHttpClient; //alcinoe
19194: 694 uPSI_ALFcnHTML; //alcinoe
19195: 695 uPSI_ALFTPClient; //alcinoe
19196: 696 uPSI_ALInternetMessageCommon; //alcinoe
19197: 697 uPSI_ALWininetHttpClient; //alcinoe
19198: 698 uPSI_ALWininetFTPClient; //alcinoe
19199: 699 uPSI_ALWinHttpWrapper; //alcinoe
19200: 700 uPSI_ALWinHttpClient; //alcinoe
19201: 701 uPSI_ALFcnWinSock; //alcinoe
19202: 702 uPSI_ALFcnSQL; //alcinoe
19203: 703 uPSI_ALFcnCGI; //alcinoe
19204: 704 uPSI_ALFcnExecute; //alcinoe
19205: 705 uPSI_ALFcnFile; //alcinoe
19206: 706 uPSI_ALFcnMime; //alcinoe
19207: 707 uPSI_ALPhpRunner; //alcinoe
19208: 708 uPSI_ALGraphic; //alcinoe
19209: 709 uPSI_ALIniFiles; //alcinoe
19210: 710 uPSI_ALMemCachedClient; //alcinoe
19211: 711 unit uPSI_MyGrids; //mX4
19212: 712 uPSI_ALMultiPartMixedParser; //alcinoe
19213: 713 uPSI_ALSMTPClient; //alcinoe
19214: 714 uPSI_ALNNTPCClient; //alcinoe
19215: 715 uPSI_ALHintBalloon; //alcinoe
19216: 716 unit uPSI_ALXmlDoc; //alcinoe
19217: 717 unit uPSI_IPCThrd; //VCL
19218: 718 unit uPSI_MonForm; //VCL
19219: 719 unit uPSI_TeCanvas; //Orpheus
19220: 720 unit uPSI_Ovcmisc; //Orpheus
19221: 721 unit uPSI_ovcfiler; //Orpheus
19222: 722 unit uPSI_ovcstate; //Orpheus
19223: 723 unit uPSI_ovccoco; //Orpheus

```

```

19224: 724 unit uPSI_ovcrvexp; //Orpheus
19225: 725 unit uPSI_OvcFormatSettings; //Orpheus
19226: 726 unit uPSI_OvcUtils; //Orpheus
19227: 727 unit uPSI_ovcstore; //Orpheus
19228: 728 unit uPSI_ovcstr; //Orpheus
19229: 729 unit uPSI_ovcmru; //Orpheus
19230: 730 unit uPSI_ovccmd; //Orpheus
19231: 731 unit uPSI_ovctimer; //Orpheus
19232: 732 unit uPSI_ovcintl; //Orpheus
19233: 733 uPSI_AfCircularBuffer; //AsyncFree
19234: 734 uPSI_AfUtils; //AsyncFree
19235: 735 uPSI_AfSafeSync; //AsyncFree
19236: 736 uPSI_AfComPortCore; //AsyncFree
19237: 737 uPSI_AfComPort; //AsyncFree
19238: 738 uPSI_AfPortControls; //AsyncFree
19239: 739 uPSI_AfDataDispatcher; //AsyncFree
19240: 740 uPSI_AfViewers; //AsyncFree
19241: 741 uPSI_AfDataTerminal; //AsyncFree
19242: 742 uPSI_SimplePortMain; //AsyncFree
19243: 743 unit uPSI_ovcclock; //Orpheus
19244: 744 unit uPSI_o32intlst; //Orpheus
19245: 745 unit uPSI_o32ledlabel; //Orpheus
19246: 746 unit uPSI_AlMySQLClient; //alcinoe
19247: 747 unit uPSI_ALFBXClient; //alcinoe
19248: 748 unit uPSI_ALFcnsSQL; //alcinoe
19249: 749 unit uPSI_AsyncTimer; //mX4
19250: 750 unit uPSI_ApplicationFileIO; //mX4
19251: 751 unit uPSI_PsAPI; //VCL6
19252: 752 uPSI_ovcuser; //Orpheus
19253: 753 uPSI_ovcurl; //Orpheus
19254: 754 uPSI_ovcvlb; //Orpheus
19255: 755 uPSI_ovccolor; //Orpheus
19256: 756 uPSI_ALFBXLib; //alcinoe
19257: 757 uPSI_ovcmeter; //Orpheus
19258: 758 uPSI_ovcpeakm; //Orpheus
19259: 759 uPSI_O32BGSty; //Orpheus
19260: 760 uPSI_ovcBidi; //Orpheus
19261: 761 uPSI_ovctcary; //Orpheus
19262: 762 uPSI_DXPUtils; //mX4
19263: 763 uPSI_ALMultiPartBaseParser; //alcinoe
19264: 764 uPSI_ALMultiPartAlternativeParser; //alcinoe
19265: 765 uPSI_ALPOP3Client; //alcinoe
19266: 766 uPSI_SmallUtils; //mX4
19267: 767 uPSI_MakeApp; //mX4
19268: 768 uPSI_O32MouseMon; //Orpheus
19269: 769 uPSI_OvcCache; //Orpheus
19270: 770 uPSI_ovccalc; //Orpheus
19271: 771 uPSI_Joystick; //OpenGL
19272: 772 uPSI_ScreenSaver; //OpenGL
19273: 773 uPSI_XCollection; //OpenGL
19274: 774 uPSI_Polynomials; //OpenGL
19275: 775 uPSI_PersistentClasses, //9.86 //OpenGL
19276: 776 uPSI_VectorLists; //OpenGL
19277: 777 uPSI_XOpenGL; //OpenGL
19278: 778 uPSI_MeshUtils; //OpenGL
19279: 779 unit uPSI_JclSysUtils; //JCL
19280: 780 unit uPSI_JclBorlandTools; //JCL
19281: 781 unit JclFileUtils_max; //JCL
19282: 782 uPSI_AfDataControls; //AsyncFree
19283: 783 uPSI_GLSilhouette; //OpenGL
19284: 784 uPSI_JclSysUtils_class; //JCL
19285: 785 uPSI_JclFileUtils_class; //JCL
19286: 786 uPSI_FileUtil; //JCL
19287: 787 uPSI_changefind; //mX4
19288: 788 uPSI_cmdIntf; //mX4
19289: 789 uPSI_fservise; //mX4
19290: 790 uPSI_Keyboard; //OpenGL
19291: 791 uPSI_VRMLParser; //OpenGL
19292: 792 uPSI_GLFileVRML; //OpenGL
19293: 793 uPSI-Octree; //OpenGL
19294: 794 uPSI_GLPolyhedron; //OpenGL
19295: 795 uPSI_GLCrossPlatform; //OpenGL
19296: 796 uPSI_GLParticles; //OpenGL
19297: 797 uPSI_GLNavigator; //OpenGL
19298: 798 uPSI_GLStarRecord; //OpenGL
19299: 799 uPSI_GLTextureCombiners; //OpenGL
19300: 800 uPSI_GLCanvas; //OpenGL
19301: 801 uPSI_GeometryBB; //OpenGL
19302: 802 uPSI_GeometryCoordinates; //OpenGL
19303: 803 uPSI_VectorGeometry; //OpenGL
19304: 804 uPSI_BumpMapping; //OpenGL
19305: 805 uPSI_TGA; //OpenGL
19306: 806 uPSI_GLVectorFileObjects; //OpenGL
19307: 807 uPSI_IMM; //VCL
19308: 808 uPSI_CategoryButtons; //VCL
19309: 809 uPSI_ButtonGroup; //VCL
19310: 810 uPSI_DbExcept; //VCL
19311: 811 uPSI_AxCtrls; //VCL
19312: 812 uPSI_GL_actorUnit1; //OpenGL

```

```

19313: 813 uPSI_StdVCL; //VCL
19314: 814 unit CurvesAndSurfaces; //OpenGL
19315: 815 uPSI_DataAwareMain; //AsyncFree
19316: 816 uPSI_TabNotBk; //VCL
19317: 817 uPSI_udwsfiler; //mX4
19318: 818 uPSI_synaip; //Synapse!
19319: 819 uPSI_synacode; //Synapse
19320: 820 uPSI_synachar; //Synapse
19321: 821 uPSI_synamisc; //Synapse
19322: 822 uPSI_synaser; //Synapse
19323: 823 uPSI_synaicnv; //Synapse
19324: 824 uPSI_tlntsend; //Synapse
19325: 825 uPSI_pingsend; //Synapse
19326: 826 uPSI_blksock; //Synapse
19327: 827 uPSI_asnlutil; //Synapse
19328: 828 uPSI_dnssend; //Synapse
19329: 829 uPSI_clamsend; //Synapse
19330: 830 uPSI_ldapsend; //Synapse
19331: 831 uPSI_mimemess; //Synapse
19332: 832 uPSI_slogsend; //Synapse
19333: 833 uPSI_mimepart; //Synapse
19334: 834 uPSI_mimeinln; //Synapse
19335: 835 uPSI_ftpsend; //Synapse
19336: 836 uPSI_ftptsend; //Synapse
19337: 837 uPSI_httpsend; //Synapse
19338: 838 uPSI_sntpsend; //Synapse
19339: 839 uPSI_smtpsend; //Synapse
19340: 840 uPSI_snmpsend; //Synapse
19341: 841 uPSI_imapsend; //Synapse
19342: 842 uPSI_pop3send; //Synapse
19343: 843 uPSI_nntpsend; //Synapse
19344: 844 uPSI_ssl_cryptlib; //Synapse
19345: 845 uPSI_ssl_openssl; //Synapse
19346: 846 uPSI_synhttp_daemon; //Synapse
19347: 847 uPSI_NetWork; //mX4
19348: 848 uPSI_PingThread; //Synapse
19349: 849 uPSI_JvThreadTimer; //JCL
19350: 850 unit uPSI_wwSystem; //InfoPower
19351: 851 unit uPSI_IdComponent; //Indy
19352: 852 unit uPSI_IdIOHandlerThrottle; //Indy
19353: 853 unit uPSI_Themes; //VCL
19354: 854 unit uPSI_StdStyleActnCtrls; //VCL
19355: 855 unit uPSI_UDDIHelper; //VCL
19356: 856 unit uPSI_IdIMAP4Server; //Indy
19357: 857 uPSI_VariantSymbolTable; //VCL //3.9.9.92
19358: 858 uPSI_udf_glob; //mX4
19359: 859 uPSI_TabGrid; //VCL
19360: 860 uPSI_JsDBTreeView; //mX4
19361: 861 uPSI_JsSendMail; //mX4
19362: 862 uPSI_dbTvRecordList; //mX4
19363: 863 uPSI_TreeVwEx; //mX4
19364: 864 uPSI_ECDataLink; //mX4
19365: 865 uPSI_dbTree; //mX4
19366: 866 uPSI_dbTreeCBox; //mX4
19367: 867 unit uPSI_Debug; //TfrmDebug
19368: 868 uPSI_TimeFnCs; //mX4
19369: 869 uPSI_FileIntf; //VCL
19370: 870 uPSI_SockTransport; //RTL
19371: 871 unit uPSI_WinInet; //RTL
19372:
19373:
19374:
19375: //////////////////////////////////////
19376: //Form Template Library FTL
19377: //////////////////////////////////////
19378:
19379: 26 FTL For Form Building out of the Script, eg. 399_form_templates.txt
19380:
19381: 045 unit uPSI_VListView TFormListView;
19382: 263 unit uPSI_JvProfiler32; TProfReport
19383: 270 unit uPSI_ImgList; TCustomImageList
19384: 278 unit uPSI_JvImageWindow; TJvImageWindow
19385: 317 unit uPSI_JvParserForm; TJvHTMLParserForm
19386: 497 unit uPSI_DebugBox; TDebugBox
19387: 513 unit uPSI_ImageWin; TImageForm, TImageForm2
19388: 514 unit uPSI_CustomDrawTreeView; TCustomDrawForm
19389: 515 unit uPSI_GraphWin; TGraphWinForm
19390: 516 unit uPSI_actionMain; TActionForm
19391: 518 unit uPSI_CtlPanel; TAppletApplication
19392: 529 unit uPSI_MDIEdit; TEditForm
19393: 535 unit uPSI_CoolMain; {browser} TWebMainForm
19394: 538 unit uPSI_frmExportMain; TSynexportForm
19395: 585 unit uPSI_usniffer; {PortScanForm} TSniffForm
19396: 600 unit uPSI_ThreadForm; TThreadSortForm;
19397: 618 unit uPSI_delphi_arduino_Unit1; TLEDForm
19398: 620 unit uPSI_fplotMain; TfplotForm1
19399: 660 unit uPSI_JvDBQueryParamsForm; TJvQueryParamsDialog
19400: 677 unit uPSI_OptionsFrm; TfrmOptions;
19401: 718 unit uPSI_MonForm; TMonitorForm

```



```

19402: 742 unit uPSI_SimplePortMain;
19403: 770 unit uPSI_ovccalc;
19404: 810 unit uPSI_DbExcept;
19405: 812 unit uPSI_GL_actorUnit1;
19406: 846 unit uPSI_synhttp_daemon;
19407: 867 unit uPSI_Debug;
19408:
19409:
19410: ex.:with TEditForm.create(self) do begin
19411:     caption:= 'Template Form Tester';
19412:     FormStyle:= fsStayOnTop;
19413:     with editor do begin
19414:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
19415:         SelStart:= 0;
19416:         Modified:= False;
19417:     end;
19418: end;
19419: with TWebMainForm.create(self) do begin
19420:     URLs.Text:= 'http://www.kleiner.ch';
19421:     URLsClick(self); Show;
19422: end;
19423: with TSynexportForm.create(self) do begin
19424:     Caption:= 'Synexport HTML RTF tester';
19425:     Show;
19426: end;
19427: with TThreadSortForm.create(self) do begin
19428:     showmodal; free;
19429: end;
19430:
19431: with TCustomDrawForm.create(self) do begin
19432:     width:=820; height:=820;
19433:     image1.height:= 600; //add properties
19434:     image1.picture.bitmap:= image2.picture.bitmap;
19435:     //SelectionBackground1Click(self) CustomDraw1Click(self);
19436:     Background1.click;
19437:     bitmap1.click;
19438:     Tile1.click;
19439:     Showmodal;
19440:     Free;
19441: end;
19442:
19443: with TfplotForm1.Create(self) do begin
19444:     BtnPlotClick(self);
19445:     Showmodal; Free;
19446: end;
19447:
19448: with TOvcCalculator.create(self) do begin
19449:     parent:= aForm;
19450:     //free;
19451:     setbounds(550,510,200,150);
19452:     displaystr:= 'maXcalc';
19453: end;
19454:
19455:
19456:
19457: //////////////////////////////////////
19458: All maXbox Tutorials Table of Content 2014
19459: //////////////////////////////////////
19460: Tutorial 00 Function-Coding (Blix the Programmer)
19461: Tutorial 01 Procedural-Coding
19462: Tutorial 02 OO-Programming
19463: Tutorial 03 Modular Coding
19464: Tutorial 04 UML Use Case Coding
19465: Tutorial 05 Internet Coding
19466: Tutorial 06 Network Coding
19467: Tutorial 07 Game Graphics Coding
19468: Tutorial 08 Operating System Coding
19469: Tutorial 09 Database Coding
19470: Tutorial 10 Statistic Coding
19471: Tutorial 11 Forms Coding
19472: Tutorial 12 SQL DB Coding
19473: Tutorial 13 Crypto Coding
19474: Tutorial 14 Parallel Coding
19475: Tutorial 15 Serial RS232 Coding
19476: Tutorial 16 Event Driven Coding
19477: Tutorial 17 Web Server Coding
19478: Tutorial 18 Arduino System Coding
19479: Tutorial 18_3 RGB LED System Coding
19480: Tutorial 19 WinCOM /Arduino Coding
19481: Tutorial 20 Regular Expressions RegEx
19482: Tutorial 21 Android Coding (coming 2013)
19483: Tutorial 22 Services Programming
19484: Tutorial 23 Real Time Systems
19485: Tutorial 24 Clean Code
19486: Tutorial 25 maXbox Configuration I+II
19487: Tutorial 26 Socket Programming with TCP
19488: Tutorial 27 XML & TreeView
19489: Tutorial 28 DLL Coding (available)
19490: Tutorial 29 UML Scripting (coming 2014)

```

```
19491:    Tutorial 30 Web of Things (coming 2014)
19492:    Tutorial 31 Closures (coming 2014)
19493:    Tutorial 32 SQL Firebird (coming 2014)
19494:
19495:
19496: ref Docu for all Type Class and Const in maXbox_types.pdf
19497: using Docu for this file is maxbox_functions_all.pdf
19498: PEP - Pascal Education Program Lib Lab
19499:
19500: http://stackoverflow.com/tags/pascalscript/hot
19501: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
19502: #locs:15162
19503: http://sourceforge.net/apps/mediawiki/maXbox
19504: http://www.blaisepascal.eu/
19505: https://github.com/maxkleiner/maXbox3.git
19506: http://www.heise.de/download/maxbox-1176464.html
19507: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
19508: https://www.facebook.com/pages/Programming-maXbox/166844836691703
19509:
19510: ---- bigbitbox code_cleared_checked----
```