```
 1: ************************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9.98 R7
 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: 22931968 V3.9.9.98 October 2014 To EKON/BASTA/JAX/IBZ/SWS/EU/DT
10: ***************Now the Funclist*****************************************
11: Funclist Function : 13307 //10766 //10165 (7648)
12: ***************Now the Proclist*****************************************
13: Proclist Procedure Size is: 8291  //6792 //6401 4752
14: ***************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1341 //995 //
16: def head:max: maXbox7: 10.09.2014 09:11:55
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: 22939! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 23150
22: ASize of EXE: 22931968 (22041600) (16586240) (13511680)
23: SHA1 Hash of maXbox 3.9.9.98: FF1FBC4C54343B301A55089CC01630434175632E
24:
25: ------------------------------------------------------------------------
26: ------------------------------------------------------------------------
27: ////////////////////////////////////////////////////////////////////////
28:
29:
30: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
31: Function *************Now the Funclist*****************
32: function  GetResStringChecked(Ident: string; const Args: array of const): string
33: Function ( Index : Longint) : Integer
34: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
35: Function _CheckAutoResult( ResultCode : HResult) : HResult
36: function _T(Name: tbtString): Variant;
37: function ABNFToText(const AText : String) : String
38: Function Abs(e : Extended) : Extended;
39: Function Ackermann( const A, B : Integer) : Integer
40: Function AcquireLayoutLock : Boolean
41: Function ActionByName( const AName : string) : TWebActionItem
42: Function ACTIVEBUFFER : PCHAR
43: Function Add : TAggregate
44: function Add : TCollectionItem
45: Function Add : TColumn
46: Function Add : TComboExItem
47: Function Add : TCookie
48: Function Add : TCoolBand
49: Function Add : TFavoriteLinkItem
50: Function Add : TFileTypeItem
51: Function Add : THeaderSection
52: Function Add : THTMLTableColumn
53: Function Add : TIdEMailAddressItem
54: Function Add : TIdMessagePart
55: Function Add : TIdUserAccount
56: Function Add : TListColumn
57: Function Add : TListItem
58: Function Add : TStatusPanel
59: Function Add : TTaskDialogBaseButtonItem
60: Function Add : TWebActionItem
61: Function Add : TWorkArea
62: Function Add( AClass : TClass) : Integer
63: Function Add( AComponent : TComponent) : Integer
64: Function Add( AItem, AData : Integer) : Integer
65: Function Add( AItem, AData : Pointer) : Pointer
66: Function Add( AItem, AData : TObject) : TObject
67: Function Add( AObject : TObject) : Integer
68: Function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
69: Function Add( const S : WideString) : Integer
70: Function Add( Image, Mask : TBitmap) : Integer
71: Function Add( Index : LongInt; const Text : string) : LongInt
72: Function Add( Sibling : TTreeNode; const S : string) : TTreeNode
73: Function Add(const S: string): Integer
74: function Add(S: string): Integer;
75: Function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
76: Function ADDCHILD : TFIELDDEF
77: Function AddChild( Index : LongInt; const Text : string) : LongInt
78: Function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
79: Function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
80: Function AddChildObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
81: Function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
82: Function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
83: Function ADDFIELDDEF : TFIELDDEF
84: Function AddFileExtIfNecessary( AFileName, AExt : string) : string
85: Function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
86: Function AddIcon( Image : TIcon) : Integer
87: Function AddImage( Value : TCustomImageList; Index : Integer) : Integer
88: Function ADDINDEXDEF : TINDEXDEF
89: Function AddItem(const Caption:String;const ImageIdx,SelectImageIdx,OverlayImagIdx,
    Indent:Int;Data:Ptr):TComboExItem
```

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

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

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

```
357: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;
358: Function ClearBit5( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
359: Function ClearBit6( const Value : Int64; const Bit : TBitRange) : Int64;
360: function CLIENTTOSCREEN(POINT:TPOINT):TPOINT
361: Function Clipboard : TClipboard
362: Function ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
363: Function ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
364: Function ClipLine( var X1, Y1, X2, Y2 : Integer; const ClipRect : TRect) : Boolean;
365: Function ClipLineToRect( var P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean
366: Function Clone( out stm : IStream) : HResult
367: Function CloneConnection : TSQLConnection
368: Function CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
369: function CLOSEQUERY:BOOLEAN
370: Function CloseVolume( var Volume : THandle) : Boolean
371: Function CloseHandle(Handle: Integer): Integer; stdcall;
372: Function CPlApplet( hwndCPl : THandle; uMsg : DWORD; lParam1, lParam2 : Longint) : Longint
373: Function CmdLine: PChar;
374: function CmdShow: Integer;
375: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
376: Function Color32( WinColor : TColor) : TColor32;
377: Function Color321( const Index : Byte; const Palette : TPalette32) : TColor32;
378: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale :BOOLean) : TColor
379: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
380: Function ColorToHTML( const Color : TColor) : String
381: function ColorToIdent(Color: Longint; var Ident: string): Boolean)
382: Function ColorToRGB(color: TColor): Longint
383: function ColorToString(Color: TColor): string)
384: Function ColorToWebColorName( Color : TColor) : string
385: Function ColorToWebColorStr( Color : TColor) : string
386: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn
387: Function Combination(npr, ncr: integer): extended;
388: Function CombinationInt(npr, ncr: integer): Int64;
389: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
390: Function CommaAdd( const AStr1, AStr2 : String) : string
391: Function CommercialRound( const X : Extended) : Int64
392: Function Commit( grfCommitFlags : Longint) : HResult
393: Function Compare( const NameExt : string) : Boolean
394: function CompareDate(const A, B: TDateTime): TValueRelationship;
395: Function CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Integer
396: Function CompareFiles(const FN1,FN2 :string; Breathe:TNotifyEvent;BreathingSender:TObject): boolean
397: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean
398: Function CompareStr( S1, S2 : string) : Integer
399: function CompareStr(const S1: string; const S2: string): Integer)
400: function CompareString(const S1: string; const S2: string): Integer)
401: Function CompareText( S1, S2 : string) : Integer
402: function CompareText(const S1: string; const S2: string): Integer)
403: Function CompareTextLike(cWildStr,cStr:string;const cWildChar:char;lCaseSensitive:boolean): boolean
404: function CompareTime(const A, B: TDateTime): TValueRelationship;
405: function CompareValueE(const A, B: Extended; Epsilon: Extended = 0): TValueRelationship; overload;
406: function CompareValueD(const A, B: Double; Epsilon: Double = 0): TValueRelationship; overload;
407: function CompareValueS(const A, B: Single; Epsilon: Single = 0): TValueRelationship; overload;
408: function CompareValueI(const A, B: Integer): TValueRelationship; overload;
409: function CompareValueI64(const A, B: Int64): TValueRelationship; overload;
410: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
411: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
412: Function ComponentTypeToString( const ComponentType : DWORD) : string
413: Function ComposeDateTime(Date,Time : TDateTime) : TDateTime;');
414: Function CompToCurrency( Value : Comp) : Currency
415: Function CompToDouble( Value : Comp) : Double
416: function ComputeFileCRC32(const FileName : String) : Integer;
417: function ComputeSHA256(astr: string; amode: char): string)  //mode F:File, S:String
418: function ComputeSHA512(astr: string; amode: char): string)  //mode F:File, S:String
419: function Concat(s: string): string
420: Function ConnectAndGetAll : string
421: Function Connected : Boolean
422: function constrain(x, a, b: integer): integer;
423: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
424: Function ConstraintsDisabled : Boolean
425: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
426: Function ContainsState( oState : TniRegularExpressionState) : boolean
427: Function ContainsStr( const AText, ASubText : string) : Boolean
428: Function ContainsText( const AText, ASubText : string) : Boolean
429: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean
430: Function Content : string
431: Function ContentFromStream( Stream : TStream) : string
432: Function ContentFromString( const S : string) : string
433: Function CONTROLSDISABLED : BOOLEAN
434: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
435: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
436: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
437: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
438: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
439: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
440: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
441: Function ConvTypeToDescription( const AType : TConvType) : string
442: Function ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
443: Function ConvTypeToFamily( const AType : TConvType) : TConvFamily;
444: Function ConvAdd(const AVal:Dbl;const AType1:TConvType;const AVal2:Dbl;const AType2,
     AResultType:TConvType): Double
```

```
445: Function ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
     AType2:TConvType): TValueRelationship
446: Function ConvDec( const AValue : Double; const AType, AAmountType : TConvType) : Double;
447: Function ConvDec1(const AValue:Dbl;const AType:TConvType;const AAmount:Dble;const
     AAmountType:TConvType):Double;
448: Function ConvDiff(const AVal1:Dbl;const AType1:TConvType;const AVal2:Dble;const AType2,
     AResuType:TConvType):Double
449: Function ConvInc( const AValue : Double; const AType, AAmountType : TConvType) : Double;
450: Function ConvInc1(const AValue:Dbl;const AType:TConvType;const AAmount:Double;const
     AAmountType:TConvType):Double;
451: Function ConvSameValue(const AValue1:Dbl;const AType1:TConvType;const AValue2:Dbl;const
     AType2:TConvType):Bool
452: Function ConvToStr( const AValue : Double; const AType : TConvType) : string
453: Function ConvWithinNext( const AValue, ATest : Double; const AType : TConvType; const AAmount : Double;
     const AAmountType : TConvType) : Boolean
454: Function ConvWithinPrevious(const AValue,ATest:Double;const AType:TConvType; const AAmount:Double;const
     AAmountType: TConvType) : Boolean
455: function Copy(s: AnyString; iFrom, iCount: Longint): AnyString;
456: Function CopyFile( Source, Dest : string; CanOverwrite : Boolean) : Boolean
457: Function CopyFileEx( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
458: Function CopyFileTo( const Source, Destination : string) : Boolean
459: function CopyFrom(Source:TStream;Count:Int64):LongInt
460: Function CopyPalette( Palette : HPALETTE) : HPALETTE
461: Function CopyTo( Length : Integer) : string
462: Function CopyTo(stm: IStream; cb: Largeint;out cbRead: Largeint;out cbWritten:Largeint): HResult
463: Function CopyToEOF : string
464: Function CopyToEOL : string
465: Function Cos(e : Extended) : Extended;
466: Function Cosecant( const X : Extended) : Extended
467: Function Cot( const X : Extended) : Extended
468: Function Cotan( const X : Extended) : Extended
469: Function CotH( const X : Extended) : Extended
470: Function Count : Integer
471: Function CountBitsCleared( X : Byte) : Integer;
472: Function CountBitsCleared1( X : Shortint) : Integer;
473: Function CountBitsCleared2( X : Smallint) : Integer;
474: Function CountBitsCleared3( X : Word) : Integer;
475: Function CountBitsCleared4( X : Integer) : Integer;
476: Function CountBitsCleared5( X : Cardinal) : Integer;
477: Function CountBitsCleared6( X : Int64) : Integer;
478: Function CountBitsSet( X : Byte) : Integer;
479: Function CountBitsSet1( X : Word) : Integer;
480: Function CountBitsSet2( X : Smallint) : Integer;
481: Function CountBitsSet3( X : ShortInt) : Integer;
482: Function CountBitsSet4( X : Integer) : Integer;
483: Function CountBitsSet5( X : Cardinal) : Integer;
484: Function CountBitsSet6( X : Int64) : Integer;
485: function countDirfiles(const apath: string): integer;
486: function CountGenerations(Ancestor,Descendent: TClass): Integer
487: Function Coversine( X : Float) : Float
488: function CRC32(const fileName: string): LongWord;
489: Function CREATEBLOBSTREAM( FIELD : TFIELD; MODE : TBLOBSTREAMMODE) : TSTREAM
490: Function CreateColumns : TDBGridColumns
491: Function CreateDataLink : TGridDataLink
492: Function CreateDir( Dir : string) : Boolean
493: function CreateDir(const Dir: string): Boolean)
494: Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
495: Function CreateEnvironmentBlock(const Options:TEnvironmentOptions;const AdditionalVars:TStrings): PChar
496: Function CREATEFIELD(OWNER:TCOMPONENT;PARENTFIELD:TOBJECTFIELD;const
     FIELDNAME:String;CREATECHILDREN:BOOL):TFIELD
497: Function CreateGlobber( sFilespec : string) : TniRegularExpression
498: Function CreateGrayMappedBmp( Handle : HBITMAP) : HBITMAP
499: Function CreateGrayMappedRes( Instance : THandle; ResName : PChar) : HBITMAP
500: function CreateGUID(out Guid: TGUID): HResult)
501: Function CreateInstance( const unkOuter : IUnknown; const iid : TGUID; out obj) : HResult
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 CreateMessageDialog(const Msg:string; DlgType:TMsgDlgType; Buttons: TMsgDlgButtons) : TForm;
505: Function CreateMessageDialog1(const
     Msg:string;DlgType:TMsgDlgType;Btns:TMsgDlgBtns;DefaultBtn:TMsgDlgBtn):TForm;
506: function CreateOleObject(const ClassName: String): IDispatch;
507: Function CREATEPARAM( FLDTYPE : TFIELDTYPE; const PARAMNAME : String; PARAMTYPE : TPARAMTYPE) : TPARAM
508: Function CreateParameter(const
     Name:WideString;DataType:TDataType;Direction:TParameterDirection;Size:Integer;Value: OleVariant):TParameter
509: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
510: Function CreateMappedBmp( Handle : HBITMAP; const OldColors, NewColors : array of TColor) : HBITMAP
511: Function CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
512: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
513: Function CreateValueBuffer( Length : Integer) : TValueBuffer
514: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
515: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
516: Function CreateRotatedFont( Font : TFont; Angle : Integer) : HFONT
517: Function CreateTwoColorsBrushPattern( Color1, Color2 : TColor) : TBitmap
518: Function CreateValueBuffer( Length : Integer) : TValueBuffer
519: Function CreateHexDump( AOwner : TWinControl) : THexDump
520: Function Csc( const X : Extended) : Extended
521: Function CscH( const X : Extended) : Extended
522: function currencyDecimals: Byte
523: function currencyFormat: Byte
```

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

```
612: Function DegToGrad( const Degrees : Extended) : Extended
613: Function DegToGrad( const Value : Extended) : Extended;
614: Function DegToGrad1( const Value : Double) : Double;
615: Function DegToGrad2( const Value : Single) : Single;
616: Function DegToRad( const Degrees : Extended) : Extended
617: Function DegToRad( const Value : Extended) : Extended;
618: Function DegToRad1( const Value : Double) : Double;
619: Function DegToRad2( const Value : Single) : Single;
620: Function DelChar( const pStr : string; const pChar : Char) : string
621: Function DelEnvironmentVar( const Name : string) : Boolean
622: Function Delete( const MsgNum : Integer) : Boolean
623: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
624: Function DeleteFile(const FileName: string): boolean)
625: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS) : Boolean
626: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
627: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
628: Function DelSpace( const pStr : string) : string
629: Function DelString( const pStr, pDelStr : string) : string
630: Function DelTree( const Path : string) : Boolean
631: Function Depth : Integer
632: Function Description : string
633: Function DescriptionsAvailable : Boolean
634: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
635: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
636: Function DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Boolean;
637: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
638: Function DialogsToPixelsX( const Dialogs : Word) : Word
639: Function DialogsToPixelsY( const Dialogs : Word) : Word
640: Function Digits( const X : Cardinal) : Integer
641: Function DirectoryExists( const Name : string) : Boolean
642: Function DirectoryExists( Directory : string) : Boolean
643: Function DiskFree( Drive : Byte) : Int64
644: function DiskFree(Drive: Byte): Int64)
645: Function DiskInDrive( Drive : Char) : Boolean
646: Function DiskSize( Drive : Byte) : Int64
647: function DiskSize(Drive: Byte): Int64)
648: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
649: Function DispatchEnabled : Boolean
650: Function DispatchMask : TMask
651: Function DispatchMethodType : TMethodType
652: Function DISPATCHPOPUP( AHANDLE : HMENU) : BOOLEAN
653: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse) : Boolean
654: Function DisplayCase( const S : String) : String
655: Function DisplayRect( Code : TDisplayCode) : TRect
656: Function DisplayRect( TextOnly : Boolean) : TRect
657: Function DisplayStream( Stream : TStream) : string
658: TBufferCoord', 'record Char : integer; Line : integer; end
659: TDisplayCoord', 'record Column : integer; Row : integer; end
660: Function DisplayCoord( AColumn, ARow : Integer) : TDisplayCoord
661: Function BufferCoord( AChar, ALine : Integer) : TBufferCoord
662: Function DomainName( const AHost : String) : String
663: Function DownloadFile(SourceFile, DestFile: string): Boolean;  //fast!
664: Function DownloadFileOpen(SourceFile, DestFile: string): Boolean;  //open process
665: Function DosPathToUnixPath( const Path : string) : string
666: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer) : Boolean;
667: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
668: Function DoubleToBcd( const AValue : Double) : TBcd;
669: Function DoubleToHex( const D : Double) : string
670: Function DoUpdates : Boolean
671: Function Dragging: Boolean;
672: Function DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
673: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect) : BOOL
674: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL
675: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UINT) : BOOL
676: {Works like InputQuery but displays 2edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
677: Function DualInputQuery(const ACapt,Prpt1,Prpt2:string;var AVal1,AVal2:string;PasswrdChar:Char= #0):Bool;
678: Function DupeString( const AText : string; ACount : Integer) : string
679: Function Edit : Boolean
680: Function EditCaption : Boolean
681: Function EditText : Boolean
682: Function EditFolderList( Folders : TStrings) : Boolean
683: Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
684: Function Elapsed( const Update : Boolean) : Cardinal
685: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
686: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
687: Function EncodeDate( Year, Month, Day : Word) : TDateTime
688: function EncodeDate(Year, Month, Day: Word): TDateTime;
689: Function EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
690: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : TDateTime
691: Function EncodeDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond: Word): TDateTime
692: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
693: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word) : TDateTime
694: Function EncodeString( s : string) : string
695: Function DecodeString( s : string) : string
696: Function EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
697: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
698: Function EndIP : String
699: Function EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
700: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
```

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

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

```
877:  FFolder: array [TJvSpecialFolder] of Integer =
878:     (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
879:      CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
880:      CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
881:      CSIDL_STARTUP, CSIDL_TEMPLATES);
882: Function FindFilesDlg(StartIn: string; SpecialFolder: TJvSpecialFolder; UseFolder: Boolean): Boolean);
883: function Findfirst(const filepath: string; attr: integer): integer;
884: function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer)
885: Function FindFirstNotOf( AFind, AText : String) : Integer
886: Function FindFirstOf( AFind, AText : String) : Integer
887: Function FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
888: Function FINDINDEXFORFIELDS( const FIELDS : String) : TINDEXDEF
889: Function FindInstanceOf( AClass : TClass; AExact : Boolean; AStartAt : Integer) : Integer
890: Function FINDITEM( VALUE : INTEGER; KIND : TFINDITEMKIND) : TMENUITEM
891: function FindItemId( Id : Integer) : TCollectionItem
892: Function FindKey( const KeyValues : array of const) : Boolean
893: Function FINDLAST : BOOLEAN
894: Function FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
895: Function FindModuleClass( AClass : TComponentClass) : TComponent
896: Function FindModuleName( const AClass : string) : TComponent
897: Function FINDNEXT : BOOLEAN
898: function FindNext: integer;
899: function FindNext2(var F: TSearchRec): Integer)
900: Function FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible : Boolean) : TTabSheet
901: Function FindNextToSelect : TTreeNode
902: Function FINDPARAM( const VALUE : String) : TPARAM
903: Function FindParam( const Value : WideString) : TParameter
904: Function FINDPRIOR : BOOLEAN
905: Function FindResource( ModuleHandle : HMODULE; ResourceName, ResourceType : PChar) : TResourceHandle
906: Function FindSession( const SessionName : string) : TSession
907: function FindStringResource(Ident: Integer): string)
908: Function FindText( const SearchStr:string;StartPos,Length: Integer; Options: TSearchTypes):Integer
909: Function FindUnusedFileName( const FileName, FileExt, Suffix : AnsiString) : AnsiString
910: function FindVCLWindow(const Pos: TPoint): TWinControl;
911: function FindWindow(C1, C2: PChar): Longint;
912: Function FindInPaths(const fileName,paths: String): String;
913: Function Finger : String
914: Function First : TClass
915: Function First : TComponent
916: Function First : TObject
917: Function FirstDelimiter( const delimiters : string; const Str : String) : integer;
918: Function FirstDelimiter1( const delimiters : WideString; const Str : WideString) : integer;
919: Function FirstInstance( const ATitle : string) : Boolean
920: Function FloatPoint( const X, Y : Float) : TFloatPoint;
921: Function FloatPoint1( const P : TPoint) : TFloatPoint;
922: Function FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) : Boolean
923: Function FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
924: Function FloatRect1( const Rect : TRect) : TFloatRect;
925: Function FloatsEqual( const X, Y : Float) : Boolean
926: Function FloatToBin(const D: Double): string;     //doubletohex -> hextobin! in buffer
927: Function FloatToCurr( Value : Extended) : Currency
928: Function FloatToDateTime( Value : Extended) : TDateTime
929: Function FloatToStr( Value : Extended) : string;
930: Function FloatToStr(e : Extended) : String;
931: Function FloatToStrF( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer) : string;
932: function FloatToStrF(Value: Extended; Format: TFloatFormat;Precision:Integer; Digits: Integer): string)
933: Function FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings
     : TFormatSettings) : string;
934: Function FloatToStrFS( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer;
     FormatSettings : TFormatSettings) : string;
935: function FloatToText(BufferArg: PChar; const Value: Extended; ValueType: TFloatValue;Format: TFloatFormat;
     Precision,Digits: Integer): Integer)
936: Function Floor( const X : Extended) : Integer
937: Function FloorInt( Value : Integer; StepSize : Integer) : Integer
938: Function FloorJ( const X : Extended) : Integer
939: Function Flush( const Count : Cardinal) : Boolean
940: Function Flush(var t: Text): Integer
941: function FmtLoadStr(Ident: Integer; const Args: array of const): string)
942: function FOCUSED:BOOLEAN
943: Function ForceBackslash( const PathName : string) : string
944: Function ForceDirectories( const Dir : string) : Boolean
945: Function ForceDirectories( Dir : string) : Boolean
946: Function ForceDirectories( Name : string) : Boolean
947: Function ForceInBox( const Point : TPoint; const Box : TRect) : TPoint
948: Function ForceInRange( A, Min, Max : Integer) : Integer
949: Function ForceInRangeR( const A, Min, Max : Double) : Double
950: Function ForEach( AProc : TBucketProc; AInfo : Pointer) : Boolean;
951: Function ForEach1( AEvent : TBucketEvent) : Boolean;
952: Function ForegroundTask: Boolean
953: function Format(const Format: string; const Args: array of const): string;
954: Function FormatBcd( const Format : string; Bcd : TBcd) : string
955: FUNCTION FormatBigInt(s: string): STRING;
956: function FormatByteSize(const bytes: int64): string;
957: function FormatBuf(var Buffer:PChar;BufLen:Card;const Format:string;FmtLen:Card;const Args:array of
     const):Cardinal
958: Function FormatCount( iCount : integer; const sSingular : string; const sPlural : string) : string
959: Function FormatCurr( Format : string; Value : Currency) : string;
960: function FormatCurr(const Format: string; Value: Currency): string)
961: Function FormatDateTime( Format : string; DateTime : TDateTime) : string;
```

```
 962: function FormatDateTime(const fmt: string; D: TDateTime): string;
 963: Function FormatFloat( Format : string; Value : Extended) : string;
 964: function FormatFloat(const Format: string; Value: Extended): string)
 965: Function FormatFloat( Format : string; Value : Extended) : string;
 966: Function FormatFloat2( Format : string; Value : Extended; FormatSettings : TFormatSettings) : string;
 967: Function FormatCurr( Format : string; Value : Currency) : string;
 968: Function FormatCurr2(Format: string; Value : Currency; FormatSettings : TFormatSettings) : string;
 969: Function Format2(const Format:string;const Args:array of const;const FSettings:TFormatSettings): string
 970: FUNCTION FormatInt(i: integer): STRING;
 971: FUNCTION FormatInt64(i: int64): STRING;
 972: Function FormatMaskText( const EditMask : string; const Value : string) : string
 973: Function FormatValue( AValue : Cardinal) : string
 974: Function FormatVersionString( const HiV, LoV : Word) : string;
 975: Function FormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
 976: function Frac(X: Extended): Extended);
 977: Function FreeResource( ResData : HGLOBAL) : LongBool
 978: Function FromCommon( const AValue : Double) : Double
 979: function FromCommon(const AValue: Double): Double;
 980: Function FTPGMTDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 981: Function FTPLocalDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 982: Function FTPMLSToGMTDateTime( const ATimeStamp : String) : TDateTime
 983: Function FTPMLSToLocalDateTime( const ATimeStamp : String) : TDateTime
 984: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
 985: //Function Funclist Size is: 6444 of mX3.9.8.9
 986: Function FutureValue(const Rate:Extended;NPeriods:Integer;const Payment,PresentValue:Extended;PaymentTime:
      TPaymentTime):Extended
 987: Function FullTimeToStr(SUMTime: TDateTime): string;');
 988: Function Gauss( const x, Spread : Double) : Double
 989: function Gauss(const x,Spread: Double): Double;
 990: Function GCD(x, y : LongInt) : LongInt;
 991: Function GCDJ( X, Y : Cardinal) : Cardinal
 992: Function GDAL: LongWord
 993: Function GdiFlush : BOOL
 994: Function GdiSetBatchLimit( Limit : DWORD) : DWORD
 995: Function GdiGetBatchLimit : DWORD
 996: Function GenerateHeader : TIdHeaderList
 997: Function GeometricMean( const X : TDynFloatArray) : Float
 998: Function Get( AURL : string) : string;
 999: Function Get2( AURL : string) : string;
1000: Function Get8087CW : Word
1001: function GetActiveOleObject(const ClassName: String): IDispatch;
1002: Function GetAliasDriverName( const AliasName : string) : string
1003: Function GetAPMBatteryFlag : TAPMBatteryFlag
1004: Function GetAPMBatteryFullLifeTime : DWORD
1005: Function GetAPMBatteryLifePercent : Integer
1006: Function GetAPMBatteryLifeTime : DWORD
1007: Function GetAPMLineStatus : TAPMLineStatus
1008: Function GetAppdataFolder : string
1009: Function GetAppDispatcher : TComponent
1010: function GetArrayLength: integer;
1011: Function GetASCII: string;
1012: Function GetASCIILine: string;
1013: Function GetAsHandle( Format : Word) : THandle
1014: function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
1015: Function GetBackupFileName( const FileName : string) : string
1016: function GetBaseAddress(PID:DWORD):DWORD;  //Process API
1017: Function GetBBitmap( Value : TBitmap) : TBitmap
1018: Function GetBIOSCopyright : string
1019: Function GetBIOSDate : TDateTime
1020: Function GetBIOSExtendedInfo : string
1021: Function GetBIOSName : string
1022: Function getBitmap(apath: string): TBitmap;
1023: Function GetBitmap( Index : Integer; Image : TBitmap) : Boolean  //object
1024: Function getBitMapObject(const bitmappath: string): TBitmap;
1025: Function GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
1026: Function GetCapsLockKeyState : Boolean
1027: function GetCaptureControl: TControl;
1028: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char) : TJclCdTrackInfo;
1029: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char) : string;
1030: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) : string
1031: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char) : string
1032: Function GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1033: Function GetClockValue : Int64
1034: function getCmdLine: PChar;
1035: function getCmdShow: Integer;
1036: function GetCPUSpeed: Double;
1037: Function GetColField( DataCol : Integer) : TField
1038: Function GetColorBlue( const Color : TColor) : Byte
1039: Function GetColorFlag( const Color : TColor) : Byte
1040: Function GetColorGreen( const Color : TColor) : Byte
1041: Function GetColorRed( const Color : TColor) : Byte
1042: Function GetComCtlVersion : Integer
1043: Function GetComPorts: TStringlist;
1044: Function GetCommonAppdataFolder : string
1045: Function GetCommonDesktopdirectoryFolder : string
1046: Function GetCommonFavoritesFolder : string
1047: Function GetCommonFilesFolder : string
1048: Function GetCommonProgramsFolder : string
1049: Function GetCommonStartmenuFolder : string
```

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

```
1139: Function GetHighLightColor( const Color : TColor; Luminance : Integer) : TColor
1140: function GetHINSTANCE: longword;
1141: Function GetHistoryFolder : string
1142: Function GetHitTestInfoAt( X, Y : Integer) : THitTests
1143: function getHMODULE: longword;
1144: Function GetHostByName(const AComputerName: String): String;
1145: Function GetHostName : string
1146: Function getHostIP: string;
1147: Function GetHotSpot : TPoint
1148: Function GetHueBitmap( Value : TBitmap) : TBitmap
1149: Function GetImageBitmap : HBITMAP
1150: Function GETIMAGELIST : TCUSTOMIMAGELIST
1151: Function GetIncome( const aNetto : Currency) : Currency
1152: Function GetIncome( const aNetto : Extended) : Extended
1153: Function GetIncome( const aNetto : Extended): Extended
1154: Function GetIncome(const aNetto : Extended) : Extended
1155: function GetIncome(const aNetto: Currency): Currency
1156: Function GetIncome2( const aNetto : Currency) : Currency
1157: Function GetIncome2( const aNetto : Currency): Currency
1158: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string) : string
1159: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN) : TINDEXDEF
1160: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet) : TIndexDef
1161: Function GetInstRes(Instance:THandle;ResType:TResType;const
      Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1162: Function
      GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1163: Function GetIntelCacheDescription( const D : Byte) : string
1164: Function GetInteractiveUserName : string
1165: Function GetInternetCacheFolder : string
1166: Function GetInternetFormattedFileTimeStamp( const AFilename : String) : String
1167: Function GetIPAddress( const HostName : string) : string
1168: Function GetIP( const HostName : string) : string
1169: Function GetIPHostByName(const AComputerName: String): String;
1170: Function GetIsAdmin: Boolean;
1171: Function GetItem( X, Y : Integer) : LongInt
1172: Function GetItemAt( X, Y : Integer) : TListItem
1173: Function GetItemHeight(Font: TFont): Integer;
1174: Function GetItemPath( Index : Integer) : string
1175: Function GetKeyFieldNames( List : TStrings) : Integer;
1176: Function GetKeyFieldNames1( List : TWideStrings) : Integer;
1177: Function GetKeyState( const VirtualKey : Cardinal) : Boolean
1178: Function GetLastChild : LongInt
1179: Function GetLastChild : TTreeNode
1180: Function GetLastDelimitedToken( const cDelim : char; const cStr : string) : string
1181: function GetLastError: Integer
1182: Function GetLAT_CONV_FACTOR: double;  //for WGS84 power(1 - 1 / 298.257223563, 2);
1183: Function GetLoader( Ext : string) : TBitmapLoader
1184: Function GetLoadFilter : string
1185: Function GetLocalComputerName : string
1186: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char) : Char
1187: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string) : string
1188: Function GetLocalUserName : string
1189: Function GetLoginUsername : WideString
1190: function getLongDayNames: string)
1191: Function GetLongHint(const hint: string): string
1192: function getLongMonthNames: string)
1193: Function GetMacAddresses( const Machine : string; const Addresses : TStrings) : Integer
1194: Function GetMainAppWndFromPid( PID : DWORD) : HWND
1195: Function GetMapX(C_form,apath: string; const Data: string): boolean; //c_form: [html/json/xml]
1196: //if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
1197: Procedure GetGEOMap(C_form,apath: string; const Data: string);
1198: Function GetMaskBitmap : HBITMAP
1199: Function GetMaxAppAddress : Integer
1200: Function GetMciErrorMessage( const MciErrNo : MCIERROR) : string
1201: Function GetMemoryLoad : Byte
1202: Function GetMIMEDefaultFileExt( const MIMEType : string) : TIdFileName
1203: Function GetMIMETypeFromFile( const AFile : string) : string
1204: Function GetMIMETypeFromFile( const AFile : TIdFileName) : string
1205: Function GetMinAppAddress : Integer
1206: Function GetModule : TComponent
1207: Function GetModuleHandle( ModuleName : PChar) : HMODULE
1208: Function GetModuleName( Module : HMODULE) : string
1209: Function GetModulePath( const Module : HMODULE) : string
1210: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;
1211: Function GetMorseID(InChar : Char): Word;');
1212: Function GetMorseString2(InChar : Char): string;');
1213: Function GetMorseLine(dots: boolean): string;');  //whole table!  {1 or dots}
1214: Function GetMorseTable(dots: boolean): string;');  //whole table!
1215: Function GetMorseSign(InChar : Char): string;');
1216: Function GetCommandLine: PChar; stdcall;
1217: Function GetMonochromeBitmap( Value : TBitmap) : TBitmap
1218: Function GetMultiN(aval: integer): string;
1219: Function GetName : String
1220: Function GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1221: Function GetNethoodFolder : string
1222: Function GetNext : TTreeNode
1223: Function GetNextChild( Value : LongInt) : LongInt
1224: Function GetNextChild( Value : TTreeNode) : TTreeNode
1225: Function GetNextDelimitedToken( const cDelim : char; var cStr : String) : String
```

```
1226: Function GetNextItem( StartItem: TListItem;Direction:TSearchDirection;States:TItemStates) : TListItem
1227: Function GetNextPacket : Integer
1228: Function getNextSibling : TTreeNode
1229: Function GetNextVisible : TTreeNode
1230: Function GetNode( ItemId : HTreeItem) : TTreeNode
1231: Function GetNodeAt( X, Y : Integer) : TTreeNode
1232: Function GetNodeDisplayWidth( Node : TOutlineNode) : Integer
1233: function GetNumberOfProcessors: longint;
1234: Function GetNumLockKeyState : Boolean
1235: Function GetObjectProperty( Instance : TPersistent; const PropName : string) : TObject
1236: Function GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1237: Function GetOptionalParam( const ParamName : string) : OleVariant
1238: Function GetOSName: string;
1239: Function GetOSVersion: string;
1240: Function GetOSNumber: string;
1241: Function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
1242: Function GetPackageModuleHandle( PackageName : PChar) : HMODULE
1243: function GetPageSize: Cardinal;
1244: Function GetParameterFileName : string
1245: Function GetParams( var OwnerData : OleVariant) : OleVariant
1246: Function GETPARENTCOMPONENT : TCOMPONENT
1247: Function GetParentForm(control: TControl): TForm
1248: Function GETPARENTMENU : TMENU
1249: Function GetPassword : Boolean
1250: Function GetPassword : string
1251: Function GetPersonalFolder : string
1252: Function GetPidFromProcessName( const ProcessName : string) : DWORD
1253: function getPI: extended;        //of const PI math
1254: Function GetPosition : TPoint
1255: Function GetPrev : TTreeNode
1256: Function GetPrevChild( Value : LongInt) : LongInt
1257: Function GetPrevChild( Value : TTreeNode) : TTreeNode
1258: Function getPrevSibling : TTreeNode
1259: Function GetPrevVisible : TTreeNode
1260: Function GetPrinthoodFolder : string
1261: Function GetPrivilegeDisplayName( const PrivilegeName : string) : string
1262: Function getProcessList: TStrings;
1263: Function GetProcessId : TIdPID
1264: Function GetProcessNameFromPid( PID : DWORD) : string
1265: Function GetProcessNameFromWnd( Wnd : HWND) : string
1266: Function GetProcessMemoryInfo(Process: THandle;ppsmemCounters: TProcessMemoryCounters;cb: DWORD):BOOL
1267: Function getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1268: Function getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1269: Function GetProgramFilesFolder : string
1270: Function GetProgramsFolder : string
1271: Function GetProxy : string
1272: Function GetQuoteChar : WideString
1273: Function GetQrCode4(Width,Height:Word; Correct_Level:string;const
      Data:string;aformat:string):TLinearBitmap;
1274: Function GetQrCodetoFile(Width,Height:Word;Correct_Level:string;const
      Data:string;aformat:string):TLinearBitmap;
1275: Function GetRate : Double
1276: Function getPerfTime: string;
1277: Function getRuntime: string;
1278: Function GetRBitmap( Value : TBitmap) : TBitmap
1279: Function GetReadableName( const AName : string) : string
1280: Function GetRecentDocs : TStringList
1281: Function GetRecentFolder : string
1282: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer) : OleVariant;
1283: Function GetRecords1(Count:Integer; out RecsOut:Integer;Options:Integer;const CommandText:WideString;var
      Params, OwnerData : OleVariant) : OleVariant;
1284: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString) : _Recordset
1285: Function GetRegisteredCompany : string
1286: Function GetRegisteredOwner : string
1287: Function GetResource(ResType:TResType;const
      Name:string;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor:Bool
1288: Function GetResourceName( ObjStream : TStream; var AName : string) : Boolean
1289: Function GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1290: Function GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1291: Function GetRValue( rgb : DWORD) : Byte
1292: Function GetGValue( rgb : DWORD) : Byte
1293: Function GetBValue( rgb : DWORD) : Byte
1294: Function GetCValue( cmyk : COLORREF) : Byte
1295: Function GetMValue( cmyk : COLORREF) : Byte
1296: Function GetYValue( cmyk : COLORREF) : Byte
1297: Function GetKValue( cmyk : COLORREF) : Byte
1298: Function CMYK( c, m, y, k : Byte) : COLORREF
1299: Function GetOSName: string;
1300: Function GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR) : FARPROC
1301: Function GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1302: Function GetSafeCallExceptionMsg : String
1303: Function GetSaturationBitmap( Value : TBitmap) : TBitmap
1304: Function GetSaveFilter : string
1305: Function GetSaver( Ext : string) : TBitmapLoader
1306: Function GetScrollLockKeyState : Boolean
1307: Function GetSearchString : string
1308: Function GetSelections( AList : TList) : TTreeNode
1309: function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1310: Function GetSendToFolder : string
```

```
1311: Function GetServer : IAppServer
1312: Function GetServerList : OleVariant
1313: Function GetShadowColor( const Color : TColor; Luminance : Integer) : TColor
1314: Function GetShellProcessHandle : THandle
1315: Function GetShellProcessName : string
1316: Function GetShellVersion : Cardinal
1317: function getShortDayNames: string)
1318: Function GetShortHint(const hint: string): string
1319: function getShortMonthNames: string)
1320: Function GetSizeOfFile( const FileName : string) : Int64;
1321: Function GetSizeOfFile1( Handle : THandle) : Int64;
1322: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1323: Function GetStartmenuFolder : string
1324: Function GetStartupFolder : string
1325: Function GetStringProperty( Instance : TPersistent; const PropName : string) : WideString
1326: Function GetSuccessor( cChar : char) : TniRegularExpressionState
1327: Function GetSwapFileSize : Integer
1328: Function GetSwapFileUsage : Integer
1329: Function GetSystemLocale : TIdCharSet
1330: Function GetSystemMetrics( nIndex : Integer) : Integer
1331: Function GetSystemPathSH(Folder: Integer): TFilename ;
1332: Function GetTableNameFromQuery( const SQL : Widestring) : Widestring
1333: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1334: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1335: Function GetTasksList( const List : TStrings) : Boolean
1336: Function getTeamViewerID: string;
1337: Function GetTemplatesFolder : string
1338: Function GetText : PwideChar
1339: function GetText:PChar
1340: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1341: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1342: Function GetTextItem( const Value : string) : Longint
1343: function GETTEXTLEN:INTEGER
1344: Function GetThreadLocale: Longint; stdcall
1345: Function GetCurrentThreadID: LongWord; stdcall;
1346: Function GetTickCount : Cardinal
1347: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal) : Cardinal
1348: Function GetTicketNr : longint
1349: Function GetTime : Cardinal
1350: Function GetTime : TDateTime
1351: Function GetTimeout : Integer
1352: Function GetTimeStr: String
1353: Function GetTimeString: String
1354: Function GetTodayFiles(startdir, amask: string): TStringlist;
1355: Function getTokenCounts : integer
1356: Function GetTotalPageFileMemory : Integer
1357: Function GetTotalPhysicalMemory : Integer
1358: Function GetTotalVirtualMemory : Integer
1359: Function GetUniqueFileName( const APath, APrefix, AExt : String) : String
1360: Function GetUseNowForDate : Boolean
1361: Function GetUserDomainName( const CurUser : string) : string
1362: Function GetUserName : string
1363: Function GetUserName: string;
1364: Function GetUserObjectName( hUserObject : THandle) : string
1365: Function GetValueBitmap( Value : TBitmap) : TBitmap
1366: Function GetValueMSec : Cardinal
1367: Function GetValueStr : String
1368: Function GetVersion: int;
1369: Function GetVersionString(FileName: string): string;
1370: Function getVideoDrivers: string;
1371: Function GetVisibleNode( Index : LongInt) : TOutlineNode
1372: Function GetVolumeFileSystem( const Drive : string) : string
1373: Function GetVolumeName( const Drive : string) : string
1374: Function GetVolumeSerialNumber( const Drive : string) : string
1375: Function GetWebAppServices : IWebAppServices
1376: Function GetWebRequestHandler : IWebRequestHandler
1377: Function GetWindowCaption( Wnd : HWND) : string
1378: Function GetWindowDC(hdwnd: HWND): HDC;
1379: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1380: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1381: Function GetWindowsComputerID : string
1382: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1383: Function GetWindowsFolder : string
1384: Function GetWindowsServicePackVersion : Integer
1385: Function GetWindowsServicePackVersionString : string
1386: Function GetWindowsSystemFolder : string
1387: Function GetWindowsTempFolder : string
1388: Function GetWindowsUserID : string
1389: Function GetWindowsVersion : TWindowsVersion
1390: Function GetWindowsVersionString : string
1391: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1392: Function GMTToLocalDateTime( S : string) : TDateTime
1393: Function GotoKey : Boolean
1394: Function GradToCycle( const Grads : Extended) : Extended
1395: Function GradToDeg( const Grads : Extended) : Extended
1396: Function GradToDeg( const Value : Extended) : Extended;
1397: Function GradToDeg1( const Value : Double) : Double;
1398: Function GradToDeg2( const Value : Single) : Single;
1399: Function GradToRad( const Grads : Extended) : Extended
```

```
1400: Function GradToRad( const Value : Extended) : Extended;
1401: Function GradToRad1( const Value : Double) : Double;
1402: Function GradToRad2( const Value : Single) : Single;
1403: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1404: Function GreenComponent( const Color32 : TColor32) : Integer
1405: function GUIDToString(const GUID: TGUID): string)
1406: Function HandleAllocated : Boolean
1407: function HandleAllocated: Boolean;
1408: Function HandleRequest : Boolean
1409: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1410: Function HarmonicMean( const X : TDynFloatArray) : Float
1411: Function HasAsParent( Value : TTreeNode) : Boolean
1412: Function HASCHILDDEFS : BOOLEAN
1413: Function HasCurValues : Boolean
1414: Function HasExtendCharacter( const s : UTF8String) : Boolean
1415: Function HasFormat( Format : Word) : Boolean
1416: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1417: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1418: Function HashValue(AStream: TStream): LongWord
1419: Function HashValue(AStream: TStream): Word
1420: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1421: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1422: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1423: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1424: Function HashValue16( const ASrc : string) : Word;
1425: Function HashValue16stream( AStream : TStream) : Word;
1426: Function HashValue32( const ASrc : string) : LongWord;
1427: Function HashValue32Stream( AStream : TStream) : LongWord;
1428: Function HasMergeConflicts : Boolean
1429: Function hasMoreTokens : boolean
1430: Function HASPARENT : BOOLEAN
1431: function HasParent: Boolean
1432: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1433: Function HasUTF8BOM( S : TStream) : boolean;
1434: Function HasUTF8BOM1( S : AnsiString) : boolean;
1435: Function Haversine( X : Float) : Float
1436: Function Head( s : string; const subs : string; var tail : string) : string
1437: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1438: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1439: function HELPJUMP(JUMPID:STRING):BOOLEAN
1440: Function HeronianMean( const a, b : Float) : Float
1441: function HexStrToStr(Value: string): string;
1442: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1443: function HexToBin2(HexNum: string): string;
1444: Function HexToDouble( const Hex : string) : Double
1445: function HexToInt(hexnum: string): LongInt;
1446: function HexToStr(Value: string): string;
1447: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1448: function Hi(vdat: word): byte;
1449: function HiByte(W: Word): Byte)
1450: function High: Int64;
1451: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1452: function HINSTANCE: longword;
1453: function HiWord(l: DWORD): Word)
1454: function HMODULE: longword;
1455: Function HourOf( const AValue : TDateTime) : Word
1456: Function HourOfTheDay( const AValue : TDateTime) : Word
1457: Function HourOfTheMonth( const AValue : TDateTime) : Word
1458: Function HourOfTheWeek( const AValue : TDateTime) : Word
1459: Function HourOfTheYear( const AValue : TDateTime) : Word
1460: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1461: Function HourSpan( const ANow, AThen : TDateTime) : Double
1462: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1463: Function HTMLDecode( const AStr : String) : String
1464: Function HTMLEncode( const AStr : String) : String
1465: Function HTMLEscape( const Str : string) : string
1466: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1467: Function HTTPDecode( const AStr : String) : string
1468: Function HTTPEncode( const AStr : String) : string
1469: Function Hypot( const X, Y : Extended) : Extended
1470: Function IBMax( n1, n2 : Integer) : Integer
1471: Function IBMin( n1, n2 : Integer) : Integer
1472: Function IBRandomString( iLength : Integer) : String
1473: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1474: Function IBStripString( st : String; CharsToStrip : String) : String
1475: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1476: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1477: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1478: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1479: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1480: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1481:  Function RandomString( iLength : Integer) : String');
1482:  Function RandomInteger( iLow, iHigh : Integer) : Integer');
1483:  Function StripString( st : String; CharsToStrip : String) : String');
1484:  Function FormatIdentifier( Dialect : Integer; Value : String) : String');
1485:  Function FormatIdentifierValue( Dialect : Integer; Value : String) : String');
1486:  Function ExtractIdentifier( Dialect : Integer; Value : String) : String');
1487:  Function QuoteIdentifier( Dialect : Integer; Value : String) : String');
1488:  Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string;
```

```
1489:  Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String;
1490:  function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1491: Function IconToBitmap( Ico : HICON) : TBitmap
1492: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1493: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1494: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean)
1495: function IdentToColor(const Ident: string; var Color: Longint): Boolean)
1496: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1497: Function IdGetDefaultCharSet : TIdCharSet
1498: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1499: Function IdPorts2 : TStringList
1500: Function IdToMib( const Id : string) : string
1501: Function IdSHA1Hash(apath: string): string;
1502: Function IdHashSHA1(apath: string): string;
1503: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1504: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1505: Function IfThenInt( AValue : Boolean; const ATrue : integer; AFalse : integer): integer;');
1506: Function IfThenDouble( AValue : Boolean; const ATrue : double; AFalse : double): double;');
1507: Function IfThenBool( AValue : Boolean; const ATrue : boolean; AFalse : boolean): boolean;');
1508: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
1509: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1510: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean) : Boolean;
1511: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1512: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer) : TDateTime
1513: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1514: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1515: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1516: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1517: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1518: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1519: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1520: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1521: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1522: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1523: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1524: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1525: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1526: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1527: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1528: Function IncludeTrailingBackslash( S : string) : string
1529: function IncludeTrailingBackslash(const S: string): string)
1530: Function IncludeTrailingPathDelimiter( const APath : string) : string
1531: Function IncludeTrailingPathDelimiter( S : string) : string
1532: function IncludeTrailingPathDelimiter(const S: string): string)
1533: Function IncludeTrailingSlash( const APath : string) : string
1534: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliSeconds : Int64) : TDateTime
1535: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64) : TDateTime
1536: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer) : TDateTime
1537: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1538: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64) : TDateTime
1539: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer) : TDateTime
1540: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer) : TDateTime
1541: Function IndexOf( AClass : TClass) : Integer
1542: Function IndexOf( AComponent : TComponent) : Integer
1543: Function IndexOf( AObject : TObject) : Integer
1544: Function INDEXOF( const ANAME : String) : INTEGER
1545: Function IndexOf( const DisplayName : string) : Integer
1546: Function IndexOf( const Item : TBookmarkStr) : Integer
1547: Function IndexOf( const S : WideString) : Integer
1548: Function IndexOf( const View : TJclFileMappingView) : Integer
1549: Function INDEXOF( FIELD : TFIELD) : INTEGER
1550: Function IndexOf( ID : LCID) : Integer
1551: Function INDEXOF( ITEM : TMENUITEM) : INTEGER
1552: Function IndexOf( Value : TListItem) : Integer
1553: Function IndexOf( Value : TTreeNode) : Integer
1554: function IndexOf(const S: string): Integer;
1555: Function IndexOfName( const Name : WideString) : Integer
1556: function IndexOfName(Name: string): Integer;
1557: Function IndexOfObject( AObject : TObject) : Integer
1558: function IndexofObject(AObject:tObject):Integer
1559: Function IndexOfTabAt( X, Y : Integer) : Integer
1560: Function IndexStr( const AText : string; const AValues : array of string) : Integer
1561: Function IndexText( const AText : string; const AValues : array of string) : Integer
1562: Function IndexOfInteger( AList : TStringList; Value : Variant) : Integer
1563: Function IndexOfFloat( AList : TStringList; Value : Variant) : Integer
1564: Function IndexOfDate( AList : TStringList; Value : Variant) : Integer
1565: Function IndexOfString( AList : TStringList; Value : Variant) : Integer
1566: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1567: Function IndyGetHostName : string
1568: Function IndyInterlockedDecrement( var I : Integer) : Integer
1569: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1570: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
1571: Function IndyInterlockedIncrement( var I : Integer) : Integer
1572: Function IndyLowerCase( const A1 : string) : string
1573: Function IndyStrToBool( const AString : String) : Boolean
1574: Function IndyUpperCase( const A1 : string) : string
1575: Function InitCommonControl( CC : Integer) : Boolean
1576: Function InitTempPath : string
1577: Function InMainThread : boolean
```

```
1578: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1579: Function Input: Text)
1580: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1581: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1582: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1583: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1584: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1585: Function InquireSignal( RtlSigNum : Integer) : TSignalState
1586: Function InRangeR( const A, Min, Max : Double) : Boolean
1587: function Insert( Index : Integer) : TCollectionItem
1588: Function Insert( Index : Integer) : TComboExItem
1589: Function Insert( Index : Integer) : THeaderSection
1590: Function Insert( Index : Integer) : TListItem
1591: Function Insert( Index : Integer) : TStatusPanel
1592: Function Insert( Index : Integer) : TWorkArea
1593: Function Insert( Index : LongInt; const Text : string) : LongInt
1594: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1595: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1596: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1597: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1598: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
1599: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1600: Function Instance : Longint
1601: function InstanceSize: Longint
1602: Function Int(e : Extended) : Extended;
1603: function Int64ToStr(i: Int64): String;
1604: Function IntegerToBcd( const AValue : Integer) : TBcd
1605: Function Intensity( const Color32 : TColor32) : Integer;
1606: Function Intensity( const R, G, B : Single) : Single;
1607: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
      FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1608: Function InterestRate(NPeriods:Integer;const Payment,PresVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1609: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1610: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1611: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1612: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1613: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1614: Function IntMibToStr( const Value : string) : string
1615: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1616: Function IntToBin( Value : cardinal) : string
1617: Function IntToHex( Value : Integer; Digits : Integer) : string;
1618: Function IntToHex(a: integer; b: integer): string;
1619: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1620: function IntToHex64(Value: Int64; Digits: Integer): string)
1621: Function IntTo3Str( Value : Longint; separator: string) : string
1622: Function inttobool( aInt : LongInt) : Boolean
1623: function IntToStr(i: Int64): String;
1624: Function IntToStr64(Value: Int64): string)
1625: function IOResult: Integer
1626: Function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1627: Function IsAccel(VK: Word; const Str: string): Boolean
1628: Function IsAddressInNetwork( Address : String) : Boolean
1629: Function IsAdministrator : Boolean
1630: Function IsAlias( const Name : string) : Boolean
1631: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1632: Function IsASCII( const AByte : Byte) : Boolean;
1633: Function IsASCIILDH( const AByte : Byte) : Boolean;
1634: Function IsAssembly(const FileName: string): Boolean;
1635: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1636: Function IsBinary(const AChar : Char) : Boolean
1637: function IsConsole: Boolean)
1638: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1639: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1640: Function IsDelphiDesignMode : boolean
1641: Function IsDelphiRunning : boolean
1642: Function IsDFAState : boolean
1643: Function IsDirectory( const FileName : string) : Boolean
1644: Function IsDomain( const S : String) : Boolean
1645: function IsDragObject(Sender: TObject): Boolean;
1646: Function IsEditing : Boolean
1647: Function ISEMPTY : BOOLEAN
1648: Function IsEqual( Value : TParameters) : Boolean
1649: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1650: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1651: Function IsFirstNode : Boolean
1652: Function IsFloatZero( const X : Float) : Boolean
1653: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1654: Function IsFormOpen(const FormName: string): Boolean;
1655: Function IsFQDN( const S : String) : Boolean
1656: Function IsGrayScale : Boolean
1657: Function IsHex( AChar : Char) : Boolean;
1658: Function IsHexString(const AString: string): Boolean;
1659: Function IsHostname( const S : String) : Boolean
1660: Function IsInfinite( const AValue : Double) : Boolean
1661: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1662: Function IsInternet: boolean;
1663: Function IsLeadChar( ACh : Char) : Boolean
1664: Function IsLeapYear( Year : Word) : Boolean
```

```
1665: function IsLeapYear(Year: Word): Boolean)
1666: function IsLibrary: Boolean)
1667: Function ISLINE : BOOLEAN
1668: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1669: Function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1670: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1671: Function IsMatch( const Pattern, Text : string) : Boolean  //Grep like RegEx
1672: Function IsMainAppWindow( Wnd : HWND) : Boolean
1673: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1674: function IsMemoryManagerSet: Boolean)
1675: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1676: function IsMultiThread: Boolean)
1677: Function IsNumeric( AChar : Char) : Boolean;
1678: Function IsNumeric2( const AString : string) : Boolean;
1679: Function IsNTFS: Boolean;
1680: Function IsOctal( AChar : Char) : Boolean;
1681: Function IsOctalString(const AString: string) : Boolean;
1682: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1683: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1684: Function IsPM( const AValue : TDateTime) : Boolean
1685: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1686: Function IsPortAvailable( ComNum : Cardinal) : Boolean');
1687: Function IsCOMPortReal( ComNum : Cardinal) : Boolean');
1688: Function IsCOM( ComNum : Cardinal) : Boolean');
1689: Function IsCOMPort: Boolean');
1690: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1691: Function IsPrimeRM( N : Cardinal) : Boolean  //rabin miller
1692: Function IsPrimeTD( N : Cardinal) : Boolean  //trial division
1693: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1694: Function ISqrt( const I : Smallint) : Smallint
1695: Function IsReadOnly(const Filename: string): boolean;
1696: Function IsRectEmpty( const Rect : TRect) : Boolean
1697: function IsRectEmpty(const Rect: TRect): Boolean)
1698: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1699: Function ISRIGHTTOLEFT : BOOLEAN
1700: function IsRightToLeft: Boolean
1701: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1702: Function ISSEQUENCED : BOOLEAN
1703: Function IsSystemModule( const Module : HMODULE) : Boolean
1704: Function IsSystemResourcesMeterPresent : Boolean
1705: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1706: Function IsToday( const AValue : TDateTime) : Boolean
1707: function IsToday(const AValue: TDateTime): Boolean;
1708: Function IsTopDomain( const AStr : string) : Boolean
1709: Function IsUTF8LeadByte( Lead : Char) : Boolean
1710: Function IsUTF8String( const s : UTF8String) : Boolean
1711: Function IsUTF8TrailByte( Lead : Char) : Boolean
1712: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1713: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1714: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1715: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1716: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1717: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1718: Function IsValidIdent( Ident : string) : Boolean
1719: function IsValidIdent1(const Ident: string; AllowDots: Boolean): Boolean)
1720: Function IsValidIP( const S : String) : Boolean
1721: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean
1722: Function IsValidPNG(stream: TStream): Boolean;
1723: Function IsValidJPEG(stream: TStream): Boolean;
1724: Function IsValidISBN( const ISBN : AnsiString) : Boolean
1725: Function IsVariantManagerSet: Boolean; //deprecated;
1726: Function IsVirtualPcGuest : Boolean;
1727: Function IsVmWareGuest : Boolean;
1728: Function IsVCLControl(Handle: HWnd): Boolean;
1729: Function IsWhiteString( const AStr : String) : Boolean
1730: Function IsWindowResponding( Wnd : HWND; Timeout : Integer) : Boolean
1731: Function IsWoW64: boolean;
1732: Function IsWin64: boolean;
1733: Function IsWow64String(var s: string): Boolean;
1734: Function IsWin64String(var s: string): Boolean;
1735: Function IsWindowsVista: boolean;
1736: Function isPowerof2(num: int64): boolean;
1737: Function powerOf2(exponent: integer): int64;
1738: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1739: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1740: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1741: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1742: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1743: Function ItemRect( Index : Integer) : TRect
1744: function ITEMRECT(INDEX:INTEGER):TRECT
1745: Function ItemWidth( Index : Integer) : Integer
1746: Function JavahashCode(val: string): Integer;
1747: Function JosephusG(n,k: integer; var graphout: string): integer;
1748: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1749: Function JustName(PathName : string) : string;  //in path and ext
1750: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1751: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1752: Function KeepAlive : Boolean
1753: Function KeysToShiftState(Keys: Word): TShiftState;
```

```
1754: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1755: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1756: Function KeyboardStateToShiftState: TShiftState; overload;
1757: Function Languages : TLanguages
1758: Function Last : TClass
1759: Function Last : TComponent
1760: Function Last : TObject
1761: Function LastDelimiter( Delimiters, S : string) : Integer
1762: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1763: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1764: Function Latitude2WGS84(lat: double): double;
1765: Function LCM(m,n:longint):longint;
1766: Function LCMJ( const X, Y : Cardinal) : Cardinal
1767: Function Ldexp( const X : Extended; const P : Integer) : Extended
1768: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1769: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1770: function Length: Integer;
1771: Procedure LetFileList(FileList: TStringlist; apath: string);
1772: function lengthmp3(mp3path: string):integer;
1773: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1774: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1775: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1776: function LineStart(Buffer, BufPos: PChar): PChar
1777: function LineStart(Buffer, BufPos: PChar): PChar)
1778: function ListSeparator: char;
1779: function Ln(x: Extended): Extended;
1780: Function LnXP1( const X : Extended) : Extended
1781: function Lo(vdat: word): byte;
1782: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1783: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1784: Function LoadFileAsString( const FileName : string) : string
1785: Function LoadFromFile( const FileName : string) : TBitmapLoader
1786: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1787: Function LoadPackage(const Name: string): HMODULE
1788: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1789: Function LoadStr( Ident : Integer) : string
1790: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1791: Function LoadWideStr( Ident : Integer) : WideString
1792: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1793: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1794: Function LockServer( fLock : LongBool) : HResult
1795: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1796: Function Log( const X : Extended) : Extended
1797: Function Log10( const X : Extended) : Extended
1798: Function Log2( const X : Extended) : Extended
1799: function LogBase10(X: Float): Float;
1800: Function LogBase2(X: Float): Float;
1801: Function LogBaseN(Base, X: Float): Float;
1802: Function LogN( const Base, X : Extended) : Extended
1803: Function LogOffOS : Boolean
1804: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1805: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1806: Function LongDateFormat: string;
1807: function LongTimeFormat: string;
1808: Function LongWordToFourChar( ACardinal : LongWord) : string
1809: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1810: Function LookupName( const name : string) : TInAddr
1811: Function LookupService( const service : string) : Integer
1812: function Low: Int64;
1813: Function LowerCase( S : string) : string
1814: Function Lowercase(s : AnyString) : AnyString;
1815: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1816: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1817: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1818: function MainInstance: longword
1819: function MainThreadID: longword
1820: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1821: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1822: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1823: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1824: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1825: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1826: Function MakeFile(const FileName: string): integer)');
1827: function MakeLong(A, B: Word): Longint)
1828: Function MakeTempFilename( const APath : String) : string
1829: Function MakeValidFileName( const Str : string) : string
1830: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1831: function MakeWord(A, B: Byte): Word)
1832: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1833: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1834: Function MapValues( Mapping : string; Value : string) : string
1835: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1836: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1837: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1838: Function MaskGetFldSeparator( const EditMask : string) : Integer
1839: Function MaskGetMaskBlank( const EditMask : string) : Char
1840: Function MaskGetMaskSave( const EditMask : string) : Boolean
1841: Function MaskIntlLiteralToChar( IChar : Char) : Char
```

```
1842: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1843: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1844: Function MaskString( Mask, Value : String) : String
1845: Function Match( const sString : string) : TniRegularExpressionMatchResul
1846: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1847: Function Matches( const Filename : string) : Boolean
1848: Function MatchesMask( const Filename, Mask : string) : Boolean
1849: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1850: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1851: Function Max( AValueOne, AValueTwo : Integer) : Integer
1852: function Max(const x,y: Integer): Integer;
1853: Function Max1( const B1, B2 : Shortint) : Shortint;
1854: Function Max2( const B1, B2 : Smallint) : Smallint;
1855: Function Max3( const B1, B2 : Word) : Word;
1856: function Max3(const x,y,z: Integer): Integer;
1857: Function Max4( const B1, B2 : Integer) : Integer;
1858: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1859: Function Max6( const B1, B2 : Int64) : Int64;
1860: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1861: Function MaxFloat( const X, Y : Float) : Float
1862: Function MaxFloatArray( const B : TDynFloatArray) : Float
1863: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1864: function MaxIntValue(const Data: array of Integer):Integer)
1865: Function MaxJ( const B1, B2 : Byte) : Byte;
1866: function MaxPath: string;
1867: function MaxValue(const Data: array of Double): Double)
1868: Function MaxCalc( const Formula : string) : Extended  //math expression parser
1869: Procedure MaxCalcF( const Formula : string);   //out to console memo2
1870: function MD5(const fileName: string): string;
1871: Function Mean( const Data : array of Double) : Extended
1872: Function Median( const X : TDynFloatArray) : Float
1873: Function Memory : Pointer
1874: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1875: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1876: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1877: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1878: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1879: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1880: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1881: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1882: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1883: Function MibToId( Mib : string) : string
1884: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1885: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1886: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1887: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1888: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1889: Procedure GetMidiOutputs( const List : TStrings)
1890: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
1891: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
1892: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1893: Function MIDINoteToStr( Note : TMIDINote) : string
1894: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1895: Procedure GetMidiOutputs( const List : TStrings)
1896: Procedure MidiOutCheck( Code : MMResult)
1897: Procedure MidiInCheck( Code : MMResult)
1898: Function MilliSecondOf( const AValue : TDateTime) : Word
1899: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1900: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1901: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1902: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1903: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1904: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
1905: Function MilliSecondOfTheYear( const AValue : TDateTime) : Int64
1906: Function MilliSecondsBetween( const ANow, AThen : TDateTime) : Int64
1907: Function MilliSecondSpan( const ANow, AThen : TDateTime) : Double
1908: Function milliToDateTime( MilliSecond : LongInt) : TDateTime');
1909: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
1910: Function millis: int64;
1911: Function Min( AValueOne, AValueTwo : Integer) : Integer
1912: Function Min1( const B1, B2 : Shortint) : Shortint;
1913: Function Min2( const B1, B2 : Smallint) : Smallint;
1914: Function Min3( const B1, B2 : Word) : Word;
1915: Function Min4( const B1, B2 : Integer) : Integer;
1916: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1917: Function Min6( const B1, B2 : Int64) : Int64;
1918: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1919: Function MinClientRect : TRect;
1920: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1921: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1922: Function MinFloat( const X, Y : Float) : Float
1923: Function MinFloatArray( const B : TDynFloatArray) : Float
1924: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1925: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
```

```
1926: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1927: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1928: Function MinIntValue( const Data : array of Integer) : Integer
1929: function MinIntValue(const Data: array of Integer):Integer)
1930: Function MinJ( const B1, B2 : Byte) : Byte;
1931: Function MinuteOf( const AValue : TDateTime) : Word
1932: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1933: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1934: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1935: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1936: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1937: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1938: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1939: Function MinValue( const Data : array of Double) : Double
1940: function MinValue(const Data: array of Double): Double)
1941: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1942: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1943: Function ModFloat( const X, Y : Float) : Float
1944: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1945: Function Modify( const Key : string; Value : Integer) : Boolean
1946: Function ModuleCacheID : Cardinal
1947: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1948: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1949: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1950: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1951: Function MonthOf( const AValue : TDateTime) : Word
1952: Function MonthOfTheYear( const AValue : TDateTime) : Word
1953: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1954: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1955: Function MonthStr( DateTime : TDateTime) : string
1956: Function MouseCoord( X, Y : Integer) : TGridCoord
1957: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1958: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1959: Function MoveNext : Boolean
1960: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1961: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1962: Function Name : string
1963: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1964: function NetworkVolume(DriveChar: Char): string
1965: Function NEWBOTTOMLINE : INTEGER
1966: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1967: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1968: Function NEWLINE : TMENUITEM
1969: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1970: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1971: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
      const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1972: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1973: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
      TMenuItem;AENABLED:BOOL): TMENUITEM
1974: Function NEWTOPLINE : INTEGER
1975: Function Next : TIdAuthWhatsNext
1976: Function NextCharIndex( S : String; Index : Integer) : Integer
1977: Function NextRecordSet : TCustomSQLDataSet
1978: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1979: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1980: Function NextToken : Char
1981: Function nextToken : WideString
1982: function NextToken:Char
1983: Function Norm( const Data : array of Double) : Extended
1984: Function NormalizeAngle( const Angle : Extended) : Extended
1985: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
1986: Function NormalizeRect( const Rect : TRect) : TRect
1987: function NormalizeRect(const Rect: TRect): TRect;
1988: Function Now : TDateTime
1989: function Now2: tDateTime
1990: Function NumProcessThreads : integer
1991: Function NumThreadCount : integer
1992: Function NthDayOfWeek( const AValue : TDateTime) : Word
1993: Function NtProductType : TNtProductType
1994: Function NtProductTypeString : string
1995: function Null: Variant;
1996: Function NullPoint : TPoint
1997: Function NullRect : TRect
1998: Function Null2Blank(aString:String):String;
1999: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2000: Function NumIP : integer
2001: function Odd(x: Longint): boolean;
2002: Function OffsetFromUTC : TDateTime
2003: Function OffsetPoint( const P, Offset : TPoint) : TPoint
2004: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
2005: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
2006: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
2007: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
2008: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
```

```
2009: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
2010: function OpenBit:Integer
2011: Function OpenDatabase : TDatabase
2012: Function OpenDatabase( const DatabaseName : string) : TDatabase
2013: Procedure OpenDir(adir: string);
2014: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
2015: Function OpenObject( Value : PChar) : Boolean;
2016: Function OpenObject1( Value : string) : Boolean;
2017: Function OpenSession( const SessionName : string) : TSession
2018: Function OpenVolume( const Drive : Char) : THandle
2019: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
2020: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
2021: Function OrdToBinary( const Value : Byte) : string;
2022: Function OrdToBinary1( const Value : Shortint) : string;
2023: Function OrdToBinary2( const Value : Smallint) : string;
2024: Function OrdToBinary3( const Value : Word) : string;
2025: Function OrdToBinary4( const Value : Integer) : string;
2026: Function OrdToBinary5( const Value : Cardinal) : string;
2027: Function OrdToBinary6( const Value : Int64) : string;
2028: Function OSCheck( RetVal : Boolean) : Boolean
2029: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
2030: Function OSIdentToString( const OSIdent : DWORD) : string
2031: Function Output: Text)
2032: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
2033: Function Owner : TCustomListView
2034: function Owner : TPersistent
2035: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
2036: Function PadL( pStr : String; pLth : integer) : String
2037: Function Padl(s : AnyString;I : longInt) : AnyString;
2038: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
2039: Function PadR( pStr : String; pLth : integer) : String
2040: Function Padr(s : AnyString;I : longInt) : AnyString;
2041: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
2042: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2043: Function Padz(s : AnyString;I : longInt) : AnyString;
2044: Function PaethPredictor( a, b, c : Byte) : Byte
2045: Function PARAMBYNAME( const VALUE : String) : TPARAM
2046: Function ParamByName( const Value : WideString) : TParameter
2047: Function ParamCount: Integer
2048: Function ParamsEncode( const ASrc : string) : string
2049: function ParamStr(Index: Integer): string)
2050: Function ParseDate( const DateStr : string) : TDateTime
2051: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2052: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2053: Function PathAddExtension( const Path, Extension : string) : string
2054: Function PathAddSeparator( const Path : string) : string
2055: Function PathAppend( const Path, Append : string) : string
2056: Function PathBuildRoot( const Drive : Byte) : string
2057: Function PathCanonicalize( const Path : string) : string
2058: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2059: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2060: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2061: Function PathEncode( const ASrc : string) : string
2062: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2063: Function PathExtractFileNameNoExt( const Path : string) : string
2064: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2065: Function PathGetDepth( const Path : string) : Integer
2066: Function PathGetLongName( const Path : string) : string
2067: Function PathGetLongName2( Path : string) : string
2068: Function PathGetShortName( const Path : string) : string
2069: Function PathIsAbsolute( const Path : string) : Boolean
2070: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2071: Function PathIsDiskDevice( const Path : string) : Boolean
2072: Function PathIsUNC( const Path : string) : Boolean
2073: Function PathRemoveExtension( const Path : string) : string
2074: Function PathRemoveSeparator( const Path : string) : string
2075: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2076: Function Peek : Pointer
2077: Function Peek : TObject
2078: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2079: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2080: function Permutation(npr, k: integer): extended;
2081: function PermutationInt(npr, k: integer): Int64;
2082: Function PermutationJ( N, R : Cardinal) : Float
2083: Function Pi : Extended;
2084: Function PiE : Extended;
2085: Function PixelsToDialogsX( const Pixels : Word) : Word
2086: Function PixelsToDialogsY( const Pixels : Word) : Word
2087: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2088: Function Point( X, Y : Integer) : TPoint
2089: function Point(X, Y: Integer): TPoint)
2090: Function PointAssign( const X, Y : Integer) : TPoint
2091: Function PointDist( const P1, P2 : TPoint) : Double;
2092: function PointDist(const P1,P2: TFloatPoint): Double;
2093: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2094: function PointDist2(const P1,P2: TPoint): Double;
2095: Function PointEqual( const P1, P2 : TPoint) : Boolean
```

```
2096: Function PointIsNull( const P : TPoint) : Boolean
2097: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2098: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2099: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2100: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2101: Function Pop : Pointer
2102: Function Pop : TObject
2103: Function PopnStdDev( const Data : array of Double) : Extended
2104: Function PopnVariance( const Data : array of Double) : Extended
2105: Function PopulationVariance( const X : TDynFloatArray) : Float
2106: function Pos(SubStr, S: AnyString): Longint;
2107: Function PosEqual( const Rect : TRect) : Boolean
2108: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2109: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2110: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2111: Function Post1( AURL : string; const ASource : TStrings) : string;
2112: Function Post2( AURL : string; const ASource : TStream) : string;
2113: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2114: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2115: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2116: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2117: Function Power( const Base, Exponent : Extended) : Extended
2118: Function PowerBig(aval, n:integer): string;
2119: Function PowerIntJ( const X : Float; N : Integer) : Float;
2120: Function PowerJ( const Base, Exponent : Float) : Float;
2121: Function PowerOffOS : Boolean
2122: Function PreformatDateString( Ps : string) : string
2123: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2124: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2125: Function Printer : TPrinter
2126: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2127: Function ProcessResponse : TIdHTTPWhatsNext
2128: Function ProduceContent : string
2129: Function ProduceContentFromStream( Stream : TStream) : string
2130: Function ProduceContentFromString( const S : string) : string
2131: Function ProgIDToClassID(const ProgID: string): TGUID;
2132: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2133: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2134: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2135: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2136: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2137: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2138: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2139: Function Push( AItem : Pointer) : Pointer
2140: Function Push( AObject : TObject) : TObject
2141: Function Put1( AURL : string; const ASource : TStream) : string;
2142: Function Pythagoras( const X, Y : Extended) : Extended
2143: Function queryDLLInterface( var queryList : TStringList) : TStringList
2144: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2145: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2146: Function queryPerformanceCounter2(mse: int64): int64;
2147: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2148: //Function QueryPerformanceFrequency(mse: int64): boolean;
2149: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2150: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2151: Procedure QueryPerformanceCounter1(var aC: Int64);
2152: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2153: Function Quote( const ACommand : String) : SmallInt
2154: Function QuotedStr( S : string) : string
2155: Function RadToCycle( const Radians : Extended) : Extended
2156: Function RadToDeg( const Radians : Extended) : Extended
2157: Function RadToDeg( const Value : Extended) : Extended;
2158: Function RadToDeg1( const Value : Double) : Double;
2159: Function RadToDeg2( const Value : Single) : Single;
2160: Function RadToGrad( const Radians : Extended) : Extended
2161: Function RadToGrad( const Value : Extended) : Extended;
2162: Function RadToGrad1( const Value : Double) : Double;
2163: Function RadToGrad2( const Value : Single) : Single;
2164: Function RandG( Mean, StdDev : Extended) : Extended
2165: function Random(const ARange: Integer): Integer;
2166: function random2(a: integer): double
2167: function RandomE: Extended;
2168: function RandomF: Extended;
2169: Function RandomFrom( const AValues : array of string) : string;
2170: Function RandomRange( const AFrom, ATo : Integer) : Integer
2171: function randSeed: longint
2172: Function RawToDataColumn( ACol : Integer) : Integer
2173: Function Read : Char
2174: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2175: function Read(Buffer:String;Count:LongInt):LongInt
2176: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2177: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2178: Function ReadCardinal( const AConvert : boolean) : Cardinal
2179: Function ReadChar : Char
2180: Function ReadClient( var Buffer, Count : Integer) : Integer
2181: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
```

```
2182: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2183: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2184: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
      Bool):Int
2185: Function ReadInteger( const AConvert : boolean) : Integer
2186: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2187: Function ReadLn : string
2188: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2189: function Readln(question: string): string;
2190: Function readm: string;  //read last line in memo2 - console!
2191: Function ReadLnWait( AFailCount : Integer) : string
2192: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2193: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2194: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2195: Function ReadString( const ABytes : Integer) : string
2196: Function ReadString( const Section, Ident, Default : string) : string
2197: Function ReadString( Count : Integer) : string
2198: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2199: Function ReadTimeStampCounter : Int64
2200: Function RebootOS : Boolean
2201: Function Receive( ATimeOut : Integer) : TReplyStatus
2202: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2203: Function ReceiveLength : Integer
2204: Function ReceiveText : string
2205: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2206: Function ReceiveSerialText: string
2207: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2208: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
      AMilliSec:Word):TDateTime
2209: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2210: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2211: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2212: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2213: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2214: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2215: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2216: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2217: Function Reconcile( const Results : OleVariant) : Boolean
2218: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2219: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2220: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2221: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2222: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2223: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2224: Function RectCenter( const R : TRect) : TPoint
2225: Function RectEqual( const R1, R2 : TRect) : Boolean
2226: Function RectHeight( const R : TRect) : Integer
2227: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2228: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2229: Function RectIntersection( const R1, R2 : TRect) : TRect
2230: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2231: Function RectIsEmpty( const R : TRect) : Boolean
2232: Function RectIsNull( const R : TRect) : Boolean
2233: Function RectIsSquare( const R : TRect) : Boolean
2234: Function RectIsValid( const R : TRect) : Boolean
2235: Function RectsAreValid( R : array of TRect) : Boolean
2236: Function RectUnion( const R1, R2 : TRect) : TRect
2237: Function RectWidth( const R : TRect) : Integer
2238: Function RedComponent( const Color32 : TColor32) : Integer
2239: Function Refresh : Boolean
2240: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2241: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2242: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2243: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2244: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2245: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2246: Function ReleaseHandle : HBITMAP
2247: Function ReleaseHandle : HENHMETAFILE
2248: Function ReleaseHandle : HICON
2249: Function ReleasePalette : HPALETTE
2250: Function RemainderFloat( const X, Y : Float) : Float
2251: Function Remove( AClass : TClass) : Integer
2252: Function Remove( AComponent : TComponent) : Integer
2253: Function Remove( AItem : Integer) : Integer
2254: Function Remove( AItem : Pointer) : Pointer
2255: Function Remove( AItem : TObject) : TObject
2256: Function Remove( AObject : TObject) : Integer
2257: Function RemoveBackslash( const PathName : string) : string
2258: Function RemoveDF( aString : String) : String  //removes thousand separator
2259: Function RemoveDir( Dir : string) : Boolean
2260: function RemoveDir(const Dir: string): Boolean)
2261: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2262: Function RemoveFileExt( const FileName : string) : string
2263: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2264: Function RenameFile( OldName, NewName : string) : Boolean
2265: function RenameFile(const OldName: string; const NewName: string): Boolean)
2266: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2267: Function ReplaceText( const AText, AFromText, AToText : string) : string
2268: Function Replicate(c : char;I : longInt) : String;
```

```
2269: Function Request : TWebRequest
2270: Function ResemblesText( const AText, AOther : string) : Boolean
2271: Function Reset : Boolean
2272: function Reset2(mypath: string):string;
2273: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2274: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2275: Function Response : TWebResponse
2276: Function ResumeSupported : Boolean
2277: Function RETHINKHOTKEYS : BOOLEAN
2278: Function RETHINKLINES : BOOLEAN
2279: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2280: Function RetrieveCurrentDir : string
2281: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2282: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2283: Function RetrieveMailBoxSize : integer
2284: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2285: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2286: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2287: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2288: Function ReverseBits( Value : Byte) : Byte;
2289: Function ReverseBits1( Value : Shortint) : Shortint;
2290: Function ReverseBits2( Value : Smallint) : Smallint;
2291: Function ReverseBits3( Value : Word) : Word;
2292: Function ReverseBits4( Value : Cardinal) : Cardinal;
2293: Function ReverseBits4( Value : Integer) : Integer;
2294: Function ReverseBits5( Value : Int64) : Int64;
2295: Function ReverseBytes( Value : Word) : Word;
2296: Function ReverseBytes1( Value : Smallint) : Smallint;
2297: Function ReverseBytes2( Value : Integer) : Integer;
2298: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2299: Function ReverseBytes4( Value : Int64) : Int64;
2300: Function ReverseString( const AText : string) : string
2301: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var
      HostName:String):Bool;
2302: Function Revert : HResult
2303: Function RGB(R,G,B: Byte): TColor;
2304: Function RGB2BGR( const Color : TColor) : TColor
2305: Function RGB2TColor( R, G, B : Byte) : TColor
2306: Function RGBToWebColorName( RGB : Integer) : string
2307: Function RGBToWebColorStr( RGB : Integer) : string
2308: Function RgbToHtml( Value : TColor) : string
2309: Function HtmlToRgb(const Value: string): TColor;
2310: Function RightStr( const AStr : String; Len : Integer) : String
2311: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2312: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2313: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2314: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2315: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2316: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2317: Function Round(e : Extended) : Longint;
2318: Function Round64(e: extended): Int64;
2319: Function RoundAt( const Value : string; Position : SmallInt) : string
2320: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2321: Function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended;');
2322: Function SimpleRoundTo(const AValue: Extended; const ADigit: TRoundToRange): Extended;');
2323: Function RoundFrequency( const Frequency : Integer) : Integer
2324: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2325: Function RoundPoint( const X, Y : Double) : TPoint
2326: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2327: Function RowCount : Integer
2328: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2329: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2330: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2331: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2332: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2333: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2334: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2335: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2336: Function S_AddBackSlash( const ADirName : string) : string
2337: Function S_AllTrim( const cStr : string) : string
2338: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2339: Function S_Cut( const cStr : string; const iLen : integer) : string
2340: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2341: Function S_DirExists( const ADir : string) : Boolean
2342: Function S_Empty( const cStr : string) : boolean
2343: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2344: Function S_LargeFontsActive : Boolean
2345: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2346: Function S_LTrim( const cStr : string) : string
2347: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2348: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2349: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2350: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2351: Function S_RTrim( const cStr : string) : string
2352: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2353: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2354: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2355: Function S_Space( const iLen : integer) : String
2356: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
```

```
2357: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2358: Function S_StrCRC32( const Text : string) : LongWORD
2359: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2360: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2361: Function S_StringtoUTF_8( const AString : string) : string
2362: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2363: function S_StrToReal(const cStr: string; var R: Double): Boolean
2364: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2365: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2366: Function S_UTF_8ToString( const AString : string) : string
2367: Function S_WBox( const AText : string) : integer
2368: Function SameDate( const A, B : TDateTime) : Boolean
2369: function SameDate(const A, B: TDateTime): Boolean;
2370: Function SameDateTime( const A, B : TDateTime) : Boolean
2371: function SameDateTime(const A, B: TDateTime): Boolean;
2372: Function SameFileName( S1, S2 : string) : Boolean
2373: Function SameText( S1, S2 : string) : Boolean
2374: function SameText(const S1: string; const S2: string): Boolean)
2375: Function SameTime( const A, B : TDateTime) : Boolean
2376: function SameTime(const A, B: TDateTime): Boolean;
2377: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2378: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2379: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2380: Function SampleVariance( const X : TDynFloatArray) : Float
2381: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2382: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2383: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2384: Function SaveToFile( const AFileName : TFileName) : Boolean
2385: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2386: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2387: Function ScanF(const aformat: String; const args: array of const): string;
2388: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2389: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2390: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2391: function SearchRecattr: integer;
2392: function SearchRecExcludeAttr: integer;
2393: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2394: function SearchRecname: string;
2395: function SearchRecsize: integer;
2396: function SearchRecTime: integer;
2397: Function Sec( const X : Extended) : Extended
2398: Function Secant( const X : Extended) : Extended
2399: Function SecH( const X : Extended) : Extended
2400: Function SecondOf( const AValue : TDateTime) : Word
2401: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2402: Function SecondOfTheHour( const AValue : TDateTime) : Word
2403: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2404: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2405: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2406: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2407: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2408: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2409: Function SectionExists( const Section : string) : Boolean
2410: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2411: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2412: function Seek(Offset:LongInt;Origin:Word):LongInt
2413: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2414: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2415: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2416: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2417: Function SendBuf( var Buf, Count : Integer) : Integer
2418: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2419: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2420: Function SendKey( AppName : string; Key : Char) : Boolean
2421: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2422: Function SendStream( AStream : TStream) : Boolean
2423: Function SendStreamThenDrop( AStream : TStream) : Boolean
2424: Function SendText( const S : string) : Integer
2425: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2426: Function SendSerialText(Data: String): cardinal
2427: Function Sent : Boolean
2428: Function ServicesFilePath: string
2429: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2430: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2431: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2432: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2433: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2434: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2435: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2436: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2437: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2438: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2439: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2440: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2441: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2442: Function SetCurrentDir( Dir : string) : Boolean
```

```
2443: function SetCurrentDir(const Dir: string): Boolean)
2444: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2445: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2446: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2447: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2448: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2449: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2450: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2451: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2452: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2453: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2454: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2455: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2456: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2457: Function SetLocalTime( Value : TDateTime) : boolean
2458: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2459: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2460: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2461: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2462: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2463: Function SetSize( libNewSize : Longint) : HResult
2464: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2465: Function Sgn( const X : Extended) : Integer
2466: function SHA1(const fileName: string): string;
2467: function SHA256(astr: string; amode: char): string)
2468: function SHA512(astr: string; amode: char): string)
2469: Function ShareMemoryManager : Boolean
2470: Function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2471: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2472: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2473: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2474: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2475: function ShortDateFormat: string;
2476: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2477: function ShortTimeFormat: string;
2478: function SHOWMODAL:INTEGER
2479: Function ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:
      TColor; BW: Integer;Title:String;BeforeShowModal:TNotifyEvent): TModalResult');
2480: Function
      ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2481: function ShowWindow(C1: HWND; C2: integer): boolean;
2482: procedure ShowMemory     //in Dialog
2483: function ShowMemory2: string;
2484: Function ShutDownOS : Boolean
2485: Function Signe( const X, Y : Extended) : Extended
2486: Function Sign( const X : Extended) : Integer
2487: Function Sin(e : Extended) : Extended;
2488: Function sinc( const x : Double) : Double
2489: Function SinJ( X : Float) : Float
2490: Function Size( const AFileName : String) : Integer
2491: function SizeOf: Longint;
2492: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2493: function SlashSep(const Path, S: String): String
2494: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2495: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2496: Function SmallPoint(X, Y: Integer): TSmallPoint)
2497: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2498: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2499: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2500: Function SoundexProc( const AText, AOther : string) : Boolean
2501: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2502: Function SoundexWord( const AText : string) : Word
2503: Function SourcePos : Longint
2504: function SourcePos:LongInt
2505: Function Split0( Str : string; const substr : string) : TStringList
2506: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2507: Function SQLRequiresParams( const SQL : WideString) : Boolean
2508: Function Sqr(e : Extended) : Extended;
2509: Function Sqrt(e : Extended) : Extended;
2510: Function StartIP : String
2511: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2512: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2513: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2514: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2515: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2516: Function StartOfAYear( const AYear : Word) : TDateTime
2517: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2518: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2519: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2520: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2521: Function StartsStr( const ASubText, AText : string) : Boolean
2522: Function StartsText( const ASubText, AText : string) : Boolean
2523: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2524: Function StartsWith( const str : string; const sub : string) : Boolean
2525: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2526: Function StatusString( StatusCode : Integer) : string
2527: Function StdDev( const Data : array of Double) : Extended
2528: Function Stop : Float
```

```
2529: Function StopCount( var Counter : TJclCounter) : Float
2530: Function StoreColumns : Boolean
2531: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2532: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2533: Function StrAlloc( Size : Cardinal) : PChar
2534: function StrAlloc(Size: Cardinal): PChar)
2535: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2536: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2537: Function StrBufSize( Str : PChar) : Cardinal
2538: function StrBufSize(const Str: PChar): Cardinal)
2539: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2540: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2541: Function StrCat( Dest : PChar; Source : PChar) : PChar
2542: function StrCat(Dest: PChar; const Source: PChar): PChar)
2543: Function StrCharLength( Str : PChar) : Integer
2544: Function StrComp( Str1, Str2 : PChar) : Integer
2545: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2546: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2547: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2548: Function Stream_to_AnsiString( Source : TStream) : ansistring
2549: Function Stream_to_Base64( Source : TStream) : ansistring
2550: Function Stream_to_decimalbytes( Source : TStream) : string
2551: Function Stream2WideString( oStream : TStream) : WideString
2552: Function StreamtoAnsiString( Source : TStream) : ansistring
2553: Function StreamToByte( Source : TStream) : string
2554: Function StreamToDecimalbytes( Source : TStream) : string
2555: Function StreamtoOrd( Source : TStream) : string
2556: Function StreamToString( Source : TStream) : string
2557: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2558: Function StrEmpty( const sString : string) : boolean
2559: Function StrEnd( Str : PChar) : PChar
2560: function StrEnd(const Str: PChar): PChar)
2561: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2562: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2563: Function StrGet(var S : String; I : Integer) : Char;
2564: Function StrGet2(S : String; I : Integer) : Char;
2565: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2566: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2567: Function StrHtmlDecode( const AStr : String) : String
2568: Function StrHtmlEncode( const AStr : String) : String
2569: Function StrToBytes(const Value: String): TBytes;
2570: Function StrIComp( Str1, Str2 : PChar) : Integer
2571: Function StringOfChar(c : char;I : longInt) : String;
2572: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2573: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2574: Function StringRefCount(const s: String): integer;
2575: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2576: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2577: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2578: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2579: Function StringToBoolean( const Ps : string) : Boolean
2580: function StringToColor(const S: string): TColor)
2581: function StringToCursor(const S: string): TCursor;
2582: function StringToGUID(const S: string): TGUID)
2583: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2584: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2585: Function StringWidth( S : string) : Integer
2586: Function StrInternetToDateTime( Value : string) : TDateTime
2587: Function StrIsDateTime( const Ps : string) : Boolean
2588: Function StrIsFloatMoney( const Ps : string) : Boolean
2589: Function StrIsInteger( const S : string) : Boolean
2590: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2591: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2592: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2593: Function StrLen( Str : PChar) : Cardinal
2594: function StrLen(const Str: PChar): Cardinal)
2595: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2596: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2597: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2598: Function StrLower( Str : PChar) : PChar
2599: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2600: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2601: Function StrNew( Str : PChar) : PChar
2602: function StrNew(const Str: PChar): PChar)
2603: Function StrNextChar( Str : PChar) : PChar
2604: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2605: Function StrParse( var sString : string; const sDelimiters : string) : string;
2606: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2607: Function StrPas( Str : PChar) : string
2608: function StrPas(const Str: PChar): string)
2609: Function StrPCopy( Dest : PChar; Source : string) : PChar
2610: function StrPCopy(Dest: PChar; const Source: string): PChar)
2611: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2612: Function StrPos( Str1, Str2 : PChar) : PChar
2613: Function StrScan(const Str: PChar; Chr: Char): PChar)
2614: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2615: Function StrToBcd( const AValue : string) : TBcd
2616: Function StrToBool( S : string) : Boolean
2617: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
```

```
2618: Function StrToCard( const AStr : String) : Cardinal
2619: Function StrToConv( AText : string; out AType : TConvType) : Double
2620: Function StrToCurr( S : string) : Currency;
2621: function StrToCurr(const S: string): Currency)
2622: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2623: Function StrToDate( S : string) : TDateTime;
2624: function StrToDate(const s: string): TDateTime;
2625: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2626: Function StrToDateTime( S : string) : TDateTime;
2627: function StrToDateTime(const S: string): TDateTime)
2628: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2629: Function StrToDay( const ADay : string) : Byte
2630: Function StrToFloat( S : string) : Extended;
2631: function StrToFloat(s: String): Extended;
2632: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2633: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2634: Function StrToFloat( S : string) : Extended;
2635: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2636: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2637: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2638: Function StrToCurr( S : string) : Currency;
2639: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2640: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2641: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2642: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2643: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2644: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2645: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2646: Function StrToDateTime( S : string) : TDateTime;
2647: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2648: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2649: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2650: Function StrToInt( S : string) : Integer
2651: function StrToInt(s: String): Longint;
2652: Function StrToInt64( S : string) : Int64
2653: function StrToInt64(s: String): int64;
2654: Function StrToInt64Def( S : string; Default : Int64) : Int64
2655: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2656: Function StrToIntDef( S : string; Default : Integer) : Integer
2657: function StrToIntDef(const S: string; Default: Integer): Integer)
2658: function StrToIntDef(s: String; def: Longint): Longint;
2659: Function StrToMonth( const AMonth : string) : Byte
2660: Function StrToTime( S : string) : TDateTime;
2661: function StrToTime(const S: string): TDateTime)
2662: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2663: Function StrToWord( const Value : String) : Word
2664: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2665: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2666: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2667: Function StrUpper( Str : PChar) : PChar
2668: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2669: Function Sum( const Data : array of Double) : Extended
2670: Function SumFloatArray( const B : TDynFloatArray) : Float
2671: Function SumInt( const Data : array of Integer) : Integer
2672: Function SumOfSquares( const Data : array of Double) : Extended
2673: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2674: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2675: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2676: Function Supports( CursorOptions : TCursorOptions) : Boolean
2677: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2678: Function SwapWord(w : word): word)
2679: Function SwapInt(i : integer): integer)
2680: Function SwapLong(L : longint): longint)
2681: Function Swap(i : integer): integer)
2682: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2683: Function SyncTime : Boolean
2684: Function SysErrorMessage( ErrorCode : Integer) : string
2685: function SysErrorMessage(ErrorCode: Integer): string)
2686: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2687: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2688: Function SysStringLen(const S: WideString): Integer; stdcall;
2689: Function TabRect( Index : Integer) : TRect
2690: Function Tan( const X : Extended) : Extended
2691: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2692: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2693: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2694: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2695: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2696: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2697: Function TenToY( const Y : Float) : Float
2698: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2699: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2700: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
```

```
2701: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2702: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2703: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2704: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2705: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2706: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2707: Function TestBits( const Value, Mask : Byte) : Boolean;
2708: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2709: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2710: Function TestBits3( const Value, Mask : Word) : Boolean;
2711: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2712: Function TestBits5( const Value, Mask : Integer) : Boolean;
2713: Function TestBits6( const Value, Mask : Int64) : Boolean;
2714: Function TestFDIVInstruction : Boolean
2715: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2716: Function TextExtent( const Text : string) : TSize
2717: function TextHeight(Text: string): Integer;
2718: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2719: Function TextStartsWith( const S, SubS : string) : Boolean
2720: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2721: Function ConvInteger(i : integer):string;
2722: Function IntegerToText(i : integer):string;
2723: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2724: function TextWidth(Text: string): Integer;
2725: Function ThreadCount : integer
2726: function ThousandSeparator: char;
2727: Function Ticks : Cardinal
2728: Function Time : TDateTime
2729: function Time: TDateTime;
2730: function TimeGetTime: int64;
2731: Function TimeOf( const AValue : TDateTime) : TDateTime
2732: function TimeSeparator: char;
2733: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2734: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2735: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2736: Function TimeToStr( DateTime : TDateTime) : string;
2737: function TimeToStr(const DateTime: TDateTime): string;
2738: Function TimeZoneBias : TDateTime
2739: Function ToCommon( const AValue : Double) : Double
2740: function ToCommon(const AValue: Double): Double;
2741: Function Today : TDateTime
2742: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2743: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2744: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2745: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2746: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2747: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2748: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2749: function TokenComponentIdent:String
2750: Function TokenFloat : Extended
2751: function TokenFloat:Extended
2752: Function TokenInt : Longint
2753: function TokenInt:LongInt
2754: Function TokenString : string
2755: function TokenString:String
2756: Function TokenSymbolIs( const S : string) : Boolean
2757: function TokenSymbolIs(S:String):Boolean
2758: Function Tomorrow : TDateTime
2759: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2760: Function ToString : string
2761: Function TotalVariance( const Data : array of Double) : Extended
2762: Function Trace2( AURL : string) : string;
2763: Function TrackMenu( Button : TToolButton) : Boolean
2764: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2765: Function TranslateURI( const URI : string) : string
2766: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2767: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2768: Function Trim( S : string) : string;
2769: Function Trim( S : WideString) : WideString;
2770: Function Trim(s : AnyString) : AnyString;
2771: Function TrimAllOf( ATrim, AText : String) : String
2772: Function TrimLeft( S : string) : string;
2773: Function TrimLeft( S : WideString) : WideString;
2774: function TrimLeft(const S: string): string)
2775: Function TrimRight( S : string) : string;
2776: Function TrimRight( S : WideString) : WideString;
2777: function TrimRight(const S: string): string)
2778: function TrueBoolStrs: array of string
2779: Function Trunc(e : Extended) : Longint;
2780: Function Trunc64(e: extended): Int64;
2781: Function TruncPower( const Base, Exponent : Float) : Float
2782: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2783: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2784: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2785: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2786: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2787: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
      AValue:TDateTime):Boolean
```

```
2788: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2789: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
      AVal:TDateTime):Bool
2790: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2791: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2792: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2793: Function TryLock : Boolean
2794: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2795: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
2796: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2797: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2798: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2799: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2800: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2801: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2802: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2803: function TryStrToBool(const S: string; out Value: Boolean): Boolean;
2804: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2805: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2806: Function TwoToY( const Y : Float) : Float
2807: Function UCS4StringToWideString( const S : UCS4String) : WideString
2808: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2809: function Unassigned: Variant;
2810: Function UndoLastChange( FollowChange : Boolean) : Boolean
2811: function UniCodeToStr(Value: string): string;
2812: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2813: Function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2814: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2815: Function UnixPathToDosPath( const Path : string) : string
2816: Function UnixToDateTime( const AValue : Int64) : TDateTime
2817: function UnixToDateTime(U: Int64): TDateTime;
2818: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2819: Function UnlockResource( ResData : HGLOBAL) : LongBool
2820: Function UnlockVolume( var Handle : THandle) : Boolean
2821: Function UnMaskString( Mask, Value : String) : String
2822: function UpCase(ch : Char ) : Char;
2823: Function UpCaseFirst( const AStr : string) : string
2824: Function UpCaseFirstWord( const AStr : string) : string
2825: Function UpdateAction( Action : TBasicAction) : Boolean
2826: Function UpdateKind : TUpdateKind
2827: Function UPDATESTATUS : TUPDATESTATUS
2828: Function UpperCase( S : string) : string
2829: Function Uppercase(s : AnyString) : AnyString;
2830: Function URLDecode( ASrc : string) : string
2831: Function URLEncode( const ASrc : string) : string
2832: Function UseRightToLeftAlignment : Boolean
2833: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2834: Function UseRightToLeftReading : Boolean
2835: Function UTF8CharLength( Lead : Char) : Integer
2836: Function UTF8CharSize( Lead : Char) : Integer
2837: Function UTF8Decode( const S : UTF8String) : WideString
2838: Function UTF8Encode( const WS : WideString) : UTF8String
2839: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2840: Function Utf8ToAnsi( const S : UTF8String) : string
2841: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2842: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2843: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2844: Function ValidParentForm(control: TControl): TForm
2845: Function Value : Variant
2846: Function ValueExists( const Section, Ident : string) : Boolean
2847: Function ValueOf( const Key : string) : Integer
2848: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2849: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2850: Function VarArrayFromStrings( Strings : TStrings) : Variant
2851: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2852: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2853: Function VarFMTBcd : TVarType
2854: Function VarFMTBcdCreate1 : Variant;
2855: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2856: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2857: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2858: Function Variance( const Data : array of Double) : Extended
2859: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2860: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2861: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2862: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2863: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2864: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2865: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2866: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2867: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2868: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2869: Function VariantNeg( const V1 : Variant) : Variant
2870: Function VariantNot( const V1 : Variant) : Variant
2871: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2872: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2873: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2874: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
```

```
2875: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2876: function VarIsEmpty(const V: Variant): Boolean;
2877: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2878: function VarIsNull(const V: Variant): Boolean;
2879: Function VarToBcd( const AValue : Variant) : TBcd
2880: function VarType(const V: Variant): TVarType;
2881: Function VarType( const V : Variant) : TVarType
2882: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2883: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2884: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2885: Function VarIsByRef( const V : Variant) : Boolean
2886: Function VarIsEmpty( const V : Variant) : Boolean
2887: Procedure VarCheckEmpty( const V : Variant)
2888: Function VarIsNull( const V : Variant) : Boolean
2889: Function VarIsClear( const V : Variant) : Boolean
2890: Function VarIsCustom( const V : Variant) : Boolean
2891: Function VarIsOrdinal( const V : Variant) : Boolean
2892: Function VarIsFloat( const V : Variant) : Boolean
2893: Function VarIsNumeric( const V : Variant) : Boolean
2894: Function VarIsStr( const V : Variant) : Boolean
2895: Function VarToStr( const V : Variant) : string
2896: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2897: Function VarToWideStr( const V : Variant) : WideString
2898: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2899: Function VarToDateTime( const V : Variant) : TDateTime
2900: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2901: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2902: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2903: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2904: Function VarSameValue( const A, B : Variant) : Boolean
2905: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2906: Function VarIsEmptyParam( const V : Variant) : Boolean
2907: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2908: Function VarIsError1( const V : Variant) : Boolean;
2909: Function VarAsError( AResult : HRESULT) : Variant
2910: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2911: Function VarIsArray( const A : Variant) : Boolean;
2912: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2913: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2914: Function VarArrayOf( const Values : array of Variant) : Variant
2915: Function VarArrayRef( const A : Variant) : Variant
2916: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2917: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2918: Function VarArrayDimCount( const A : Variant) : Integer
2919: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2920: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2921: Function VarArrayLock( const A : Variant) : ___Pointer
2922: Procedure VarArrayUnlock( const A : Variant)
2923: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2924: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2925: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2926: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2927: Function Unassigned : Variant
2928: Function Null : Variant
2929: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2930: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2931: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2932: function VectorDot(const V1,V2: TFloatPoint): Double;
2933: Function VectorLengthSqr( const V : TFloatPoint) : Double
2934: function VectorLengthSqr(const V: TFloatPoint): Double;
2935: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2936: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2937: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2938: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2939: Function Verify( AUserName : String) : String
2940: Function Versine( X : Float) : Float
2941: function VersionCheck: boolean;
2942: function VersionCheckAct: string;
2943: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2944: Function VersionLanguageName( const LangId : Word) : string
2945: Function VersionResourceAvailable( const FileName : string) : Boolean
2946: Function Visible : Boolean
2947: function VolumeID(DriveChar: Char): string
2948: Function WaitFor( const AString : string) : string
2949: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2950: Function WaitFor1 : TWaitResult;
2951: Function WaitForData( Timeout : Longint) : Boolean
2952: Function WebColorNameToColor( WebColorName : string) : TColor
2953: Function WebColorStrToColor( WebColor : string) : TColor
2954: Function WebColorToRGB( WebColor : Integer) : Integer
2955: Function wGet(aURL, afile: string): boolean;'
2956: Function wGet2(aURL, afile: string): boolean;'  //without file open
2957: Function wGetX(aURL, afile: string): boolean;
2958: Function wGetX2(aURL, afile: string): boolean;'  //without file open
2959: Function WebGet(aURL, afile: string): boolean;'
2960: Function WebExists: boolean;   //alias to isinternet
2961: Function WeekOf( const AValue : TDateTime) : Word
2962: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2963: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
```

```
2964: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2965: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2966: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2967: Function WeeksInAYear( const AYear : Word) : Word
2968: Function WeeksInYear( const AValue : TDateTime) : Word
2969: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2970: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2971: Function WideCat( const x, y : WideString) : WideString
2972: Function WideCompareStr( S1, S2 : WideString) : Integer
2973: function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
2974: Function WideCompareText( S1, S2 : WideString) : Integer
2975: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
2976: Function WideCopy( const src : WideString; index, count : Integer) : WideString
2977: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
2978: Function WideEqual( const x, y : WideString) : Boolean
2979: Function WideFormat(const Format: WideString; const Args: array of const): WideString)
2980: Function WideGreater( const x, y : WideString) : Boolean
2981: Function WideLength( const src : WideString) : Integer
2982: Function WideLess( const x, y : WideString) : Boolean
2983: Function WideLowerCase( S : WideString) : WideString
2984: function WideLowerCase(const S: WideString): WideString)
2985: Function WidePos( const src, sub : WideString) : Integer
2986: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
2987: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
2988: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
2989: Function WideSameStr( S1, S2 : WideString) : Boolean
2990: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
2991: Function WideSameText( S1, S2 : WideString) : Boolean
2992: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
2993: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
2994: Function WideStringToUCS4String( const S : WideString) : UCS4String
2995: Function WideUpperCase( S : WideString) : WideString
2996: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
2997: function Win32Check(RetVal: boolean): boolean)
2998: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
2999: Function Win32RestoreFile( const FileName : string) : Boolean
3000: Function Win32Type : TIdWin32Type
3001: Function WinColor( const Color32 : TColor32) : TColor
3002: function winexec(FileName: pchar; showCmd: integer): integer;
3003: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3004: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3005: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
3006: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
3007: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
3008: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
3009: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
3010: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
3011: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
3012: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
3013: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
3014: Function WordToStr( const Value : Word) : String
3015: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
3016: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
3017: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
3018: Function WorkArea : Integer
3019: Function WrapText( Line : string; MaxCol : Integer) : string;
3020: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
3021: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3022: function Write(Buffer:String;Count:LongInt):LongInt
3023: Function WriteClient( var Buffer, Count : Integer) : Integer
3024: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
3025: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
3026: Function WriteString( const AString : string) : Boolean
3027: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3028: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
3029: Function wsprintf( Output : PChar; Format : PChar) : Integer
3030: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3031: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3032: Function XorDecode( const Key, Source : string) : string
3033: Function XorEncode( const Key, Source : string) : string
3034: Function XorString( const Key, Src : ShortString) : ShortString
3035: Function Yield : Bool
3036: Function YearOf( const AValue : TDateTime) : Word
3037: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3038: Function YearSpan( const ANow, AThen : TDateTime) : Double
3039: Function Yesterday : TDateTime
3040: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3041: Function( const Name : string; Proc : TUserFunction)
3042: Function using Special_Scholz from 3.8.5.0
3043: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3044: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3045: Function FloatToTime2Dec(value:Extended):Extended;
3046: Function MinToStd(value:Extended):Extended;
3047: Function MinToStdAsString(value:Extended):String;
3048: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3049: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3050: Function Round2Dec (zahl:Extended):Extended;
3051: Function GetAngle(x,y:Extended):Double;
3052: Function AddAngle(a1,a2:Double):Double;
```

```
3053:
3054: ********************************************************************
3055: unit uPSI_StText;
3056: ********************************************************************
3057: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3058: Function TextFileSize( var F : TextFile) : LongInt
3059: Function TextPos( var F : TextFile) : LongInt
3060: Function TextFlush( var F : TextFile) : Boolean
3061:
3062: ********************************************************************
3063: from  JvVCLUtils;
3064: ********************************************************************
3065: { Windows resources (bitmaps and icons) VCL-oriented routines }
3066: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3067: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
      DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3068: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
      Bitmap: TBitmap; TransparentColor: TColor);
3069: function MakeBitmap(ResID: PChar): TBitmap;
3070: function MakeBitmapID(ResID: Word): TBitmap;
3071: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3072: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3073: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3074: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3075:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3076: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3077: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3078: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3079: {$IFDEF WIN32}
3080: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3081:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3082: {$ENDIF}
3083: function MakeIcon(ResID: PChar): TIcon;
3084: function MakeIconID(ResID: Word): TIcon;
3085: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3086: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3087: {$IFDEF WIN32}
3088: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3089: {$ENDIF}
3090: { Service routines }
3091: procedure NotImplemented;
3092: procedure ResourceNotFound(ResID: PChar);
3093: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3094: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3095: function PaletteColor(Color: TColor): Longint;
3096: function WidthOf(R: TRect): Integer;
3097: function HeightOf(R: TRect): Integer;
3098: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3099: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3100: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3101: procedure Delay(MSecs: Longint);
3102: procedure CenterControl(Control: TControl);
3103: Function PaletteEntries( Palette : HPALETTE) : Integer
3104: Function WindowClassName( Wnd : HWND) : string
3105: Function ScreenWorkArea : TRect
3106: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3107: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3108: Procedure ActivateWindow( Wnd : HWND)
3109: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3110: Procedure CenterWindow( Wnd : HWND)
3111: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3112: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3113: Function DialogsToPixelsX( Dlgs : Word) : Word
3114: Function DialogsToPixelsY( Dlgs : Word) : Word
3115: Function PixelsToDialogsX( Pixs : Word) : Word
3116: Function PixelsToDialogsY( Pixs : Word) : Word
3117: {$IFDEF WIN32}
3118: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3119: function MakeVariant(const Values: array of Variant): Variant;
3120: {$ENDIF}
3121: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3122: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3123: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3124: {$IFDEF CBUILDER}
3125: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3126: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3127: {$ELSE}
3128: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3129: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3130: {$ENDIF CBUILDER}
3131: function IsForegroundTask: Boolean;
3132: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3133: function GetAveCharSize(Canvas: TCanvas): TPoint;
3134: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3135: procedure FreeUnusedOle;
3136: procedure Beep;
3137: function GetWindowsVersionJ: string;
3138: function LoadDLL(const LibName: string): THandle;
3139: function RegisterServer(const ModuleName: string): Boolean;
```

```
{$IFNDEF WIN32}
function IsLibrary: Boolean;
{$ENDIF}
{ Gradient filling routine }
type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
TFillDirection; Colors: Byte);
{ String routines }
function GetEnvVar(const VarName: string): string;
function AnsiUpperFirstChar(const S: string): string;
function StringToPChar(var S: string): PChar;
function StrPAlloc(const S: string): PChar;
procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
function DropT(const S: string): string;
{ Memory routines }
function AllocMemo(Size: Longint): Pointer;
function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
procedure FreeMemo(var fpBlock: Pointer);
function GetMemoSize(fpBlock: Pointer): Longint;
function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
{$IFNDEF COMPILER5_UP}
procedure FreeAndNil(var Obj);
{$ENDIF}
// from PNGLoader
Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
Procedure TransformRGB2LOCO( Image : TLinearBitmap)
Procedure TransformLOCO2RGB( Image : TLinearBitmap)
Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
  Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
  Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
 AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
 //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
 //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
 //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
 Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
 Procedure SetImeName( Name : TImeName)
 Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
 Function Imm32GetContext( hWnd : HWND) : HIMC
 Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
 Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
 Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
 Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
// Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
 //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
 Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
 Function Imm32IsIME( hKl : longword) : Boolean
 Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
 Procedure DragDone( Drop : Boolean)


//****************************************added  from jvjvclutils
function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
function IsPositiveResult(Value: TModalResult): Boolean;
function IsNegativeResult(Value: TModalResult): Boolean;
function IsAbortResult(const Value: TModalResult): Boolean;
function StripAllFromResult(const Value: TModalResult): TModalResult;
// returns either BrightColor or DarkColor depending on the luminance of AColor
// This function gives the same result (AFAIK) as the function used in Windows to
// calculate the desktop icon text color based on the desktop background color
function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);

procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
  const State: TOwnerDrawState; const Text: string; var Width: Integer;
  CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
  var LinkName: string; Scale: Integer = 100); overload;
procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
  const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
  CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
  var LinkName: string; Scale: Integer = 100); overload;
function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
  const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
  const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
  Scale: Integer = 100): string;
function HTMLPlainText(const Text: string): string;
function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
  const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
  const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
function HTMLPrepareText(const Text: string): string;

****************************************** uPSI_JvAppUtils;
Function GetDefaultSection( Component : TComponent) : string
```

```
3227: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3228: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3229: Function GetDefaultIniName : string
3230:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3231: Function GetDefaultIniRegKey : string
3232: Function FindForm( FormClass : TFormClass) : TForm
3233: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3234: Function ShowDialog( FormClass : TFormClass) : Boolean
3235:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3236: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3237: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3238: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3239: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3240: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3241: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3242: Function StrToIniStr( const Str : string) : string
3243: Function IniStrToStr( const Str : string) : string
3244: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3245: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3246: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3247: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3248: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3249: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
3250: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3251: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3252: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3253: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3254: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3255: Procedure AppTaskbarIcons( AppOnly : Boolean)
3256: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3257: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3258: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3259: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3260: ****************************************** uPSI_JvDBUtils;
3261: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3262: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3263: Procedure RefreshQuery( Query : TDataSet)
3264: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3265: Function DataSetSectionName( DataSet : TDataSet) : string
3266: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string)
3267: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3268: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
      Options: TLocateOptions) : Boolean
3269: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3270: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3271: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3272: Function ConfirmDelete : Boolean
3273: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3274: Procedure CheckRequiredField( Field : TField)
3275: Procedure CheckRequiredFields( const Fields : array of TField)
3276: Function DateToSQL( Value : TDateTime) : string
3277: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3278: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3279: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
      HighEmpty:Double;Inclusive:Bool):string
3280: Function StrMaskSQL( const Value : string) : string
3281: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3282: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3283: Procedure _DBError( const Msg : string)
3284:  Const('TrueExpr','String '0=0
3285:  Const('sdfStandard16','String '''''''mm''/''dd''/''yyyy''''''
3286:  Const('sdfStandard32','String ''''''''dd/mm/yyyy''''''''
3287:  Const('sdfOracle','String '''TO_DATE(''''dd/mm/yyyy''''', ''DD/MM/YYYY'')'
3288:  Const('sdfInterbase','String '''CAST(''''mm"/"dd"/"yyyy''''' AS DATE)'
3289:  Const('sdfMSSQL','String '''CONVERT(datetime, ''''mm"/"dd"/"yyyy''''', 103)'
3290:  AddTypeS('Largeint', 'Longint
3291:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3292:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3293:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3294:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3295:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3296: (*----------------------------------------------------------------------------*)
3297: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3298: begin
3299:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3300:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3301:  Function JIniReadString( const FileName, Section, Line : string) : string
3302:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3303:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3304:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3305:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3306:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3307: end;
3308:
3309: (* === compile-time registration functions === *)
3310: (*----------------------------------------------------------------------------*)
3311: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3312: begin
3313:  'UnixTimeStart','LongInt'( 25569);
```

```
3314:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3315:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3316:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3317:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3318:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3319:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3320:  Function DayOfDate( const DateTime : TDateTime) : Integer
3321:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3322:  Function YearOfDate( const DateTime : TDateTime) : Integer
3323:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3324:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3325:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3326:  Function HourOfTime( const DateTime : TDateTime) : Integer
3327:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3328:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3329:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3330:  Function IsISOLongYear( const Year : Word) : Boolean;
3331:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3332:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3333:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3334:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3335:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3336:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3337:  Function JIsLeapYear( const Year : Integer) : Boolean;
3338:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3339:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3340:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3341:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3342:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3343:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3344:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3345:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3346:  Function HoursToMSecs( Hours : Integer) : Integer
3347:  Function MinutesToMSecs( Minutes : Integer) : Integer
3348:  Function SecondsToMSecs( Seconds : Integer) : Integer
3349:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3350:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3351:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3352:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3353:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3354:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3355:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3356:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3357:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3358:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3359:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3360:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3361:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3362:  Function DosDateTimeToStr( DateTime : Integer) : string
3363:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3364:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3365:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3366:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3367:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3368:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3369:  Function FileTimeToStr( const FileTime : TFileTime) : string
3370:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3371:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3372:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3373:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3374:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3375:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3376:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3377:   TJclUnixTime32', 'Longword
3378:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3379:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3380:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3381:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3382:  Function JNullStamp : TTimeStamp
3383:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3384:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3385:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3386:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3387:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3388:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3389:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3390:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3391:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3392:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3393:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3394:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3395:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3396:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3397:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3398:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3399:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3400:   FindClass('TOBJECT'),'EJclDateTimeError
3401: end;
3402:
```

```
3403: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3404: begin
3405:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3406:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3407:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3408:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3409:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3410:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3411:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3412:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3413:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3414:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3415:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3416:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3417:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3418:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3419:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3420:  Function AbortShutDown : Boolean;
3421:  Function AbortShutDown1( const MachineName : string) : Boolean;
3422:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3423:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3424:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3425:   FindClass('TOBJECT'),'EJclCreateProcessError
3426:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3427:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
      Environment:PChar);
3428:   // with Add(EJclCreateProcessError) do
3429:  end;
3430:
3431:
3432: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3433: begin
3434:   //'AnsiSigns','Set').SetSet(['-', '+']);
3435:  'C1_UPPER','LongWord( $0001);
3436:  'C1_LOWER','LongWord( $0002);
3437:  'C1_DIGIT','LongWord').SetUInt( $0004);
3438:  'C1_SPACE','LongWord').SetUInt( $0008);
3439:  'C1_PUNCT','LongWord').SetUInt( $0010);
3440:  'C1_CNTRL','LongWord').SetUInt( $0020);
3441:  'C1_BLANK','LongWord').SetUInt( $0040);
3442:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3443:  'C1_ALPHA','LongWord').SetUInt( $0100);
3444:   AnsiChar', 'Char
3445:  Function StrIsAlpha( const S : AnsiString) : Boolean
3446:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3447:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3448:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3449:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3450:  Function StrIsDigit( const S : AnsiString) : Boolean
3451:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3452:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3453:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3454:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3455:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3456:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3457:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3458:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3459:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3460:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3461:  Function StrEscapedToString( const S : AnsiString) : AnsiString
3462:  Function JStrLower( const S : AnsiString) : AnsiString
3463:  Procedure StrLowerInPlace( var S : AnsiString)
3464:  ///Procedure StrLowerBuff( S : PAnsiChar)
3465:  Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3466:  Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3467:  Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3468:  Function StrProper( const S : AnsiString) : AnsiString
3469:  //Procedure StrProperBuff( S : PAnsiChar)
3470:  Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3471:  Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3472:  Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3473:  Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3474:  Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3475:  Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3476:  Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3477:  Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3478:  Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3479:  Function StrReverse( const S : AnsiString) : AnsiString
3480:  Procedure StrReverseInPlace( var S : AnsiString)
3481:  Function StrSingleQuote( const S : AnsiString) : AnsiString
3482:  Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
3483:  Function StrStringToEscaped( const S : AnsiString) : AnsiString
3484:  Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3485:  Function StrToHex( const Source : AnsiString) : AnsiString
3486:  Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3487:  Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3488:  Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3489:  Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3490:  Function StrTrimQuotes( const S : AnsiString) : AnsiString
```

```
3491:  Function JStrUpper( const S : AnsiString) : AnsiString
3492:  Procedure StrUpperInPlace( var S : AnsiString)
3493:  //Procedure StrUpperBuff( S : PAnsiChar)
3494:  Function StrOemToAnsi( const S : AnsiString) : AnsiString
3495:  Function StrAnsiToOem( const S : AnsiString) : AnsiString
3496:  Procedure StrAddRef( var S : AnsiString)
3497:  Function StrAllocSize( const S : AnsiString) : Longint
3498:  Procedure StrDecRef( var S : AnsiString)
3499:  //Function StrLen( S : PAnsiChar) : Integer
3500:  Function StrLength( const S : AnsiString) : Longint
3501:  Function StrRefCount( const S : AnsiString) : Longint
3502:  Procedure StrResetLength( var S : AnsiString)
3503:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3504:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3505:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3506:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3507:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3508:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3509:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3510:  Function StrFillChar(const C: Char; Count: Integer): string)');
3511:  Function IntFillChar(const I: Integer; Count: Integer): string)');
3512:  Function ByteFillChar(const B: Byte; Count: Integer): string)');
3513:  Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;');
3514:  Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3515:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3516:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3517:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3518:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
3519:  Function StrIPos( const SubStr, S : AnsiString) : Integer
3520:  Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3521:  Function StrLastPos( const SubStr, S : AnsiString) : Integer
3522:  Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3523:  Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3524:  Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3525:  Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3526:  Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3527:  Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3528:  //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3529:  //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3530:  Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3531:  Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3532:  Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3533:  Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
3534:  Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3535:  Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3536:  Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3537:  Function CharIsAlpha( const C : AnsiChar) : Boolean
3538:  Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3539:  Function CharIsBlank( const C : AnsiChar) : Boolean
3540:  Function CharIsControl( const C : AnsiChar) : Boolean
3541:  Function CharIsDelete( const C : AnsiChar) : Boolean
3542:  Function CharIsDigit( const C : AnsiChar) : Boolean
3543:  Function CharIsLower( const C : AnsiChar) : Boolean
3544:  Function CharIsNumberChar( const C : AnsiChar) : Boolean
3545:  Function CharIsPrintable( const C : AnsiChar) : Boolean
3546:  Function CharIsPunctuation( const C : AnsiChar) : Boolean
3547:  Function CharIsReturn( const C : AnsiChar) : Boolean
3548:  Function CharIsSpace( const C : AnsiChar) : Boolean
3549:  Function CharIsUpper( const C : AnsiChar) : Boolean
3550:  Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3551:  Function CharType( const C : AnsiChar) : Word
3552:  Function CharHex( const C : AnsiChar) : Byte
3553:  Function CharLower( const C : AnsiChar) : AnsiChar
3554:  Function CharUpper( const C : AnsiChar) : AnsiChar
3555:  Function CharToggleCase( const C : AnsiChar) : AnsiChar
3556:  Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3557:  Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3558:  Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer) : Integer
3559:  Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar) : Integer
3560:  Procedure StrIToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3561:  Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3562:  Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3563:  Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)
3564:  Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)
3565:  Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)
3566:  Function AddStringToStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3567:  Function BooleanToStr( B : Boolean) : AnsiString
3568:  Function FileToString( const FileName : AnsiString) : AnsiString
3569:  Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean)
3570:  Function StrToken( var S : AnsiString; Separator : AnsiChar) : AnsiString
3571:  Procedure StrTokens( const S : AnsiString; const List : TStrings)
3572:  Procedure StrTokenToStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings)
3573:  //Function StrWord( var S : PAnsiChar; out Word : AnsiString) : Boolean
3574:  Function StrToFloatSafe( const S : AnsiString) : Float
3575:  Function StrToIntSafe( const S : AnsiString) : Integer
3576:  Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer);
3577:  Function ArrayOf( List : TStrings) : TDynStringArray;
3578:   EJclStringError', 'EJclError
3579:  function IsClass(Address: TObject): Boolean;
```

```
3580:  function IsObject(Address: TObject): Boolean;
3581:  // Console Utilities
3582:  //function ReadKey: Char;
3583:  function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3584:  function JclGUIDToString(const GUID: TGUID): string;
3585:  function JclStringToGUID(const S: string): TGUID;
3586:
3587: end;
3588:
3589:
3590: ****************************************** uPSI_JvDBUtil;
3591: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const
      Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3592: Function GetQueryResult( const DatabaseName, SQL : string) : Variant
3593: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant;
      const AResultName : string) : Variant
3594: //Function StrFieldDesc( Field : FLDDesc) : string
3595: Function Var2Type( V : Variant; const VarType : Integer) : Variant
3596: Procedure CopyRecord( DataSet : TDataSet)
3597: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string;
      MasterField : Word; ModOp, DelOp : RINTQual)
3598: Procedure AddMasterPassword( Table : TTable; pswd : string)
3599: Procedure PackTable( Table : TTable)
3600: Procedure PackEncryptedTable( Table : TTable; pswd : string)
3601: Function EncodeQuotes( const S : string) : string
3602: Function Cmp( const S1, S2 : string) : Boolean
3603: Function SubStr( const S : string; const Index : Integer; const Separator : string) : string
3604: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string) : string
3605: Function ReplaceString( S : string; const OldPattern, NewPattern : string) : string
3606: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer)
3607: ******************************************uPSI_JvJvBDEUtils;***************
3608: //JvBDEUtils
3609: Function CreateDbLocate : TJvLocateObject
3610:  //Function CheckOpen( Status : DBIResult) : Boolean
3611: Procedure FetchAllRecords( DataSet : TBDEDataSet)
3612: Function TransActive( Database : TDatabase) : Boolean
3613: Function AsyncQrySupported( Database : TDatabase) : Boolean
3614: Function GetQuoteChar( Database : TDatabase) : string
3615: Procedure ExecuteQuery( const DbName, QueryText : string)
3616: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string)
3617: Function FieldLogicMap( FldType : TFieldType) : Integer
3618: Function FieldSubtypeMap( FldType : TFieldType) : Integer Value : string; Buffer : Pointer)
3619: Function GetAliasPath( const AliasName : string) : string
3620: Function IsDirectory( const DatabaseName : string) : Boolean
3621: Function GetBdeDirectory : string
3622: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean
3623: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3624: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3625: Function DataSetRecNo( DataSet : TDataSet) : Longint
3626: Function DataSetRecordCount( DataSet : TDataSet) : Longint
3627: Function DataSetPositionStr( DataSet : TDataSet) : string
3628: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)
3629: Function CurrentRecordDeleted( DataSet : TBDEDataSet) : Boolean
3630: Function IsFilterApplicable( DataSet : TDataSet) : Boolean
3631: Function IsBookmarkStable( DataSet : TBDEDataSet) : Boolean
3632: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3633: Procedure RestoreIndex( Table : TTable)
3634: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3635: Procedure PackTable( Table : TTable)
3636: Procedure ReindexTable( Table : TTable)
3637: Procedure BdeFlushBuffers
3638: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer
3639: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3640: Procedure DbNotSupported
3641: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3642: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3643: Procedure
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3644: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3645: ******************************************uPSI_JvDateUtil;
3646: function CurrentYear: Word;
3647: function IsLeapYear(AYear: Integer): Boolean;
3648: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3649: function FirstDayOfPrevMonth: TDateTime;
3650: function LastDayOfPrevMonth: TDateTime;
3651: function FirstDayOfNextMonth: TDateTime;
3652: function ExtractDay(ADate: TDateTime): Word;
3653: function ExtractMonth(ADate: TDateTime): Word;
3654: function ExtractYear(ADate: TDateTime): Word;
3655: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3656: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3657: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3658: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3659: function ValidDate(ADate: TDateTime): Boolean;
3660: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3661: function MonthsBetween(Date1, Date2: TDateTime): Double;
3662: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
```

```
3663: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3664: function DaysBetween(Date1, Date2: TDateTime): Longint;
3665: { The same as previous but if Date2 < Date1 result = 0 }
3666: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3667: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3668: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3669: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3670: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3671: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3672: { String to date conversions }
3673: function GetDateOrder(const DateFormat: string): TDateOrder;
3674: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3675: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3676: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3677: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3678: function DefDateFormat(FourDigitYear: Boolean): string;
3679: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3680: ------------------------------------------------------------------------------
3681: ******************************** JvUtils;*****************************
3682: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3683: function GetWordOnPos(const S: string; const P: Integer): string;
3684: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3685: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3686: { SubStr returns substring from string, S, separated with Separator string}
3687: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3688: { SubStrEnd same to previous function but Index numerated from the end of string }
3689: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3690: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3691: function SubWord(P: PChar; var P2: PChar): string;
3692: { NumberByWord returns the text representation of
3693:   the number, N, in normal russian language. Was typed from Monitor magazine }
3694: function NumberByWord(const N: Longint): string;
3695: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3696:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3697: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3698: { GetXYByPos is same to previous function, but returns X position in line too}
3699: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3700: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3701: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3702: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3703: function ConcatSep(const S, S2, Separator: string): string;
3704: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3705: function ConcatLeftSep(const S, S2, Separator: string): string;
3706: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3707: function MinimizeString(const S: string; const MaxLen: Integer): string;
3708: { Next 4 function for russian chars transliterating.
3709:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3710: procedure Dos2Win(var S: string);
3711: procedure Win2Dos(var S: string);
3712: function Dos2WinRes(const S: string): string;
3713: function Win2DosRes(const S: string): string;
3714: function Win2Koi(const S: string): string;
3715: { Spaces returns string consists on N space chars }
3716: function Spaces(const N: Integer): string;
3717: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3718: function AddSpaces(const S: string; const N: Integer): string;
3719: { function LastDate for russian users only } { returns date relative to current date: '' }
3720: function LastDate(const Dat: TDateTime): string;
3721: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3722: function CurrencyToStr(const Cur: currency): string;
3723: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3724: function Cmp(const S1, S2: string): Boolean;
3725: { StringCat add S2 string to S1 and returns this string }
3726: function StringCat(var S1: string; S2: string): string;
3727: { HasChar returns True, if Char, Ch, contains in string, S }
3728: function HasChar(const Ch: Char; const S: string): Boolean;
3729: function HasAnyChar(const Chars: string; const S: string): Boolean;
3730: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3731: function CountOfChar(const Ch: Char; const S: string): Integer;
3732: function DefStr(const S: string; Default: string): string;
3733: {**** files routines}
3734: { GetWinDir returns Windows folder name }
3735: function GetWinDir: TFileName;
3736: function GetSysDir: String;
3737: { GetTempDir returns Windows temporary folder name }
3738: function GetTempDir: string;
3739: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3740: function GenTempFileName(FileName: string): string;
3741: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3742: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3743: { ClearDir clears folder Dir }
3744: function ClearDir(const Dir: string): Boolean;
3745: { DeleteDir clears and than delete folder Dir }
3746: function DeleteDir(const Dir: string): Boolean;
3747: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3748: function FileEquMask(FileName, Mask: TFileName): Boolean;
3749: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3750:   Masks must be separated with comma (';') }
3751: function FileEquMasks(FileName, Masks: TFileName): Boolean;
```

```
3752: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3753: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3754: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3755: { FileGetInfo fills SearchRec record for specified file attributes}
3756: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3757: { HasSubFolder returns True, if folder APath contains other folders }
3758: function HasSubFolder(APath: TFileName): Boolean;
3759: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3760: function IsEmptyFolder(APath: TFileName): Boolean;
3761: { AddSlash add slash Char to Dir parameter, if needed }
3762: procedure AddSlash(var Dir: TFileName);
3763: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3764: function AddSlash2(const Dir: TFileName): string;
3765: { AddPath returns FileName with Path, if FileName not contain any path }
3766: function AddPath(const FileName, Path: TFileName): TFileName;
3767: function AddPaths(const PathList, Path: string): string;
3768: function ParentPath(const Path: TFileName): TFileName;
3769: function FindInPath(const FileName, PathList: string): TFileName;
3770: function FindInPaths(const fileName,paths: String): String;
3771: {$IFNDEF BCB1}
3772: { BrowseForFolder displays Browse For Folder dialog }
3773: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3774: {$ENDIF BCB1}
3775: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3776: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3777: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3778: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3779:
3780: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3781: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3782: { HasParam returns True, if program running with specified parameter, Param }
3783: function HasParam(const Param: string): Boolean;
3784: function HasSwitch(const Param: string): Boolean;
3785: function Switch(const Param: string): string;
3786: { ExePath returns ExtractFilePath(ParamStr(0)) }
3787: function ExePath: TFileName;
3788: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3789: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3790: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3791: {**** Graphic routines }
3792: { TTFontSelected returns True, if True Type font is selected in specified device context }
3793: function TTFontSelected(const DC: HDC): Boolean;
3794: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3795: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3796: {**** Windows routines }
3797: { SetWindowTop put window to top without recreating window }
3798: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3799: {**** other routines }
3800: { KeyPressed returns True, if Key VK is now pressed }
3801: function KeyPressed(VK: Integer): Boolean;
3802: procedure SwapInt(var Int1, Int2: Integer);
3803: function IntPower(Base, Exponent: Integer): Integer;
3804: function ChangeTopException(E: TObject): TObject;
3805: function StrToBool(const S: string): Boolean;
3806: {$IFNDEF COMPILER3_UP}
3807: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3808:   Length of MaxLen bytes. The compare operation is controlled by the
3809:   current Windows locale. The return value is the same as for CompareStr. }
3810: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3811: function AnsiStrIComp(S1, S2: PChar): Integer;
3812: {$ENDIF}
3813: function Var2Type(V: Variant; const VarType: Integer): Variant;
3814: function VarToInt(V: Variant): Integer;
3815: function VarToFloat(V: Variant): Double;
3816: { following functions are not documented because they are don't work properly , so don't use them }
3817: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3818: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3819: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3820: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3821: function GetParameter: string;
3822: function GetLongFileName(FileName: string): string;
3823: {* from  FileCtrl}
3824: function DirectoryExists(const Name: string): Boolean;
3825: procedure ForceDirectories(Dir: string);
3826: {# from  FileCtrl}
3827: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3828: function GetComputerID: string;
3829: function GetComputerName: string;
3830: {**** string routines }
3831: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
     same Index.Also see RAUtilsW.ReplaceSokr1 function }
3832: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3833: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3834:   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 }
3835: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3836: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
```

```
3837: function CountOfLines(const S: string): Integer;
3838: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3839: procedure DeleteEmptyLines(Ss: TStrings);
3840: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3841:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3842: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3843: {**** files routines - }
3844: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3845:   Resource can be compressed using MS Compress program}
3846: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3847: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3848: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3849: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3850: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3851: { IniReadSection read section, Section, from ini-file,
3852:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3853:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3854: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3855: { LoadTextFile load text file, FileName, into string }
3856: function LoadTextFile(const FileName: TFileName): string;
3857: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3858: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3859: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3860: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3861: {$IFDEF COMPILER3_UP}
3862: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3863: function TargetFileName(const FileName: TFileName): TFileName;
3864: { return filename ShortCut linked to }
3865: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3866: {$ENDIF COMPILER3_UP}
3867: {**** Graphic routines - }
3868: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3869: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3870: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3871: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3872: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3873: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3874: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3875: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3876: { Cinema draws some visual effect }
3877: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3878: { Roughed fills rect with special 3D pattern }
3879: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3880: { 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 }
3881: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3882: { TextWidth calculate text with for writing using standard desktop font }
3883: function TextWidth(AStr: string): Integer;
3884: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3885: function DefineCursor(Identifer: PChar): TCursor;
3886: {**** other routines - }
3887: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3888: function FindFormByClass(FormClass: TFormClass): TForm;
3889: function FindFormByClassName(FormClassName: string): TForm;
3890: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3891:   having Tag property value, equaled to Tag parameter }
3892: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3893: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3894: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3895: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3896: function RBTag(Parent: TWinControl): Integer;
3897: { AppMinimized returns True, if Application is minimized }
3898: function AppMinimized: Boolean;
3899: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3900:   if Caption parameter = '', it replaced with Application.Title }
3901: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3902: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3903:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3904: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3905:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3906: { Delay stop program execution to MSec msec }
3907: procedure Delay(MSec: Longword);
3908: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3909: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3910: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3911: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3912: function PanelBorder(Panel: TCustomPanel): Integer;
3913: function Pixels(Control: TControl; APixels: Integer): Integer;
3914: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3915: procedure Error(const Msg: string);
3916: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3917:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3918:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3919: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3920:   const HideSelColor: Boolean): string;
3921: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3922:   const HideSelColor: Boolean): Integer;
3923: function ItemHtPlain(const Text: string): string;
3924: { ClearList - clears list of TObject }
```

```
3925: procedure ClearList(List: TList);
3926: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3927: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3928: { RTTI support }
3929: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3930: function GetPropStr(Obj: TObject; const PropName: string): string;
3931: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3932: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3933: procedure PrepareIniSection(SS: TStrings);
3934: { following functions are not documented because they are don't work properly, so don't use them }
3935: {$IFDEF COMPILER2}
3936: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3937: {$ENDIF}
3938:
3939: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3940: begin
3941:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3942:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3943:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
3944:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3945:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3946:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3947:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3948:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3949: end;
3950:
3951: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3952: begin
3953:  Const('MaxInitStrNum','LongInt'( 9);
3954: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
      : array of AnsiString; MaxSplit : Integer) : Integer
3955: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
      string; MaxSplit : Integer) : Integer
3956: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
      QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3957: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3958: Function JvStrStrip( S : string) : string
3959: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3960: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3961: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3962: Function StrEatWhiteSpace( const S : string) : string
3963: Function HexToAscii( const S : AnsiString) : AnsiString
3964: Function AsciiToHex( const S : AnsiString) : AnsiString
3965: Function StripQuotes( const S1 : AnsiString) : AnsiString
3966: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3967: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3968: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3969: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3970: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3971: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
3972: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
3973: Function JvValidIdentifier( S1 : String) : Boolean
3974: Function JvEndChar( X : AnsiChar) : Boolean
3975: Procedure JvGetToken( S1, S2 : PAnsiChar)
3976: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
3977: Function IsKeyword( S1 : PAnsiChar) : Boolean
3978: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
3979: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
3980: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
3981: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
3982: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
3983: Function GetTokenCount : Integer
3984: Procedure ResetTokenCount
3985: end;
3986:
3987: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3988: begin
3989:   SIRegister_TJvQueryParamsDialog(CL);
3990:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
3991: end;
3992:
3993: ******************************** JvStrUtil /  JvStrUtils;****************************
3994: function FindNotBlankCharPos(const S: string): Integer;
3995: function AnsiChangeCase(const S: string): string;
3996: function GetWordOnPos(const S: string; const P: Integer): string;
3997: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3998: function Cmp(const S1, S2: string): Boolean;
3999: { Spaces returns string consists on N space chars }
4000: function Spaces(const N: Integer): string;
4001: { HasChar returns True, if char, Ch, contains in string, S }
4002: function HasChar(const Ch: Char; const S: string): Boolean;
4003: function HasAnyChar(const Chars: string; const S: string): Boolean;
4004: { SubStr returns substring from string, S, separated with Separator string}
4005: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
4006: { SubStrEnd same to previous function but Index numerated from the end of string }
4007: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4008: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
4009: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
```

```
4010: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4011: { GetXYByPos is same to previous function, but returns X position in line too}
4012: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4013: { AddSlash returns string with added slash char to Dir parameter, if needed }
4014: function AddSlash2(const Dir: TFileName): string;
4015: { AddPath returns FileName with Path, if FileName not contain any path }
4016: function AddPath(const FileName, Path: TFileName): TFileName;
4017: { ExePath returns ExtractFilePath(ParamStr(0)) }
4018: function ExePath: TFileName;
4019: function LoadTextFile(const FileName: TFileName): string;
4020: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4021: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4022: function ConcatSep(const S, S2, Separator: string): string;
4023: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4024: function FileEquMask(FileName, Mask: TFileName): Boolean;
4025: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4026:   Masks must be separated with comma (';') }
4027: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4028: function StringEndsWith(const Str, SubStr: string): Boolean;
4029: function ExtractFilePath2(const FileName: string): string;
4030: function StrToOem(const AnsiStr: string): string;
4031: { StrToOem translates a string from the Windows character set into the OEM character set. }
4032: function OemToAnsiStr(const OemStr: string): string;
4033: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4034: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4035: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4036: function ReplaceStr(const S, Srch, Replace: string): string;
4037: { Returns string with every occurrence of Srch string replaced with Replace string. }
4038: function DelSpace(const S: string): string;
4039: { DelSpace return a string with all white spaces removed. }
4040: function DelChars(const S: string; Chr: Char): string;
4041: { DelChars return a string with all Chr characters removed. }
4042: function DelBSpace(const S: string): string;
4043: { DelBSpace trims leading spaces from the given string. }
4044: function DelESpace(const S: string): string;
4045: { DelESpace trims trailing spaces from the given string. }
4046: function DelRSpace(const S: string): string;
4047: { DelRSpace trims leading and trailing spaces from the given string. }
4048: function DelSpace1(const S: string): string;
4049: { DelSpace1 return a string with all non-single white spaces removed. }
4050: function Tab2Space(const S: string; Numb: Byte): string;
4051: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4052: function NPos(const C: string; S: string; N: Integer): Integer;
4053: { NPos searches for a N-th position of substring C in a given string. }
4054: function MakeStr(C: Char; N: Integer): string;
4055: function MS(C: Char; N: Integer): string;
4056: { MakeStr return a string of length N filled with character C. }
4057: function AddChar(C: Char; const S: string; N: Integer): string;
4058: { AddChar return a string left-padded to length N with characters C. }
4059: function AddCharR(C: Char; const S: string; N: Integer): string;
4060: { AddCharR return a string right-padded to length N with characters C. }
4061: function LeftStr(const S: string; N: Integer): string;
4062: { LeftStr return a string right-padded to length N with blanks. }
4063: function RightStr(const S: string; N: Integer): string;
4064: { RightStr return a string left-padded to length N with blanks. }
4065: function CenterStr(const S: string; Len: Integer): string;
4066: { CenterStr centers the characters in the string based upon the Len specified. }
4067: function CompStr(const S1, S2: string): Integer;
4068: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4069: function CompText(const S1, S2: string): Integer;
4070: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4071: function Copy2Symb(const S: string; Symb: Char): string;
4072: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4073: function Copy2SymbDel(var S: string; Symb: Char): string;
4074: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4075: function Copy2Space(const S: string): string;
4076: { Copy2Symb returns a substring of a string S from begining to first white space. }
4077: function Copy2SpaceDel(var S: string): string;
4078: { Copy2SpaceDel returns a substring of a string S from begining to first
4079:   white space and removes this substring from S. }
4080: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4081: { Returns string, with the first letter of each word in uppercase,
4082:   all other letters in lowercase. Words are delimited by WordDelims. }
4083: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4084: { WordCount given a set of word delimiters, returns number of words in S. }
4085: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4086: { Given a set of word delimiters, returns start position of N'th word in S. }
4087: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4088: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4089: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4090: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4091:   delimiters, return the N'th word in S. }
4092: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4093: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4094: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4095: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4096: function QuotedString(const S: string; Quote: Char): string;
4097: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4098: function ExtractQuotedString(const S: string; Quote: Char): string;
```

```
4099: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4100:    and reduces pairs of Quote characters within the quoted string to a single character. }
4101: function FindPart(const HelpWilds, InputStr: string): Integer;
4102: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4103: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4104: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4105: function XorString(const Key, Src: ShortString): ShortString;
4106: function XorEncode(const Key, Source: string): string;
4107: function XorDecode(const Key, Source: string): string;
4108: { ** Command line routines ** }
4109: {$IFNDEF COMPILER4_UP}
4110: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4111: {$ENDIF}
4112: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4113: { ** Numeric string handling routines ** }
4114: function Numb2USA(const S: string): string;
4115: { Numb2USA converts numeric string S to USA-format. }
4116: function Dec2Hex(N: Longint; A: Byte): string;
4117: function D2H(N: Longint; A: Byte): string;
4118: { Dec2Hex converts the given value to a hexadecimal string representation
4119:   with the minimum number of digits (A) specified. }
4120: function Hex2Dec(const S: string): Longint;
4121: function H2D(const S: string): Longint;
4122: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4123: function Dec2Numb(N: Longint; A, B: Byte): string;
4124: { Dec2Numb converts the given value to a string representation with the
4125:   base equal to B and with the minimum number of digits (A) specified. }
4126: function Numb2Dec(S: string; B: Byte): Longint;
4127: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4128: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4129: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4130: function IntToRoman(Value: Longint): string;
4131: { IntToRoman converts the given value to a roman numeric string representation. }
4132: function RomanToInt(const S: string): Longint;
4133: { RomanToInt converts the given string to an integer value. If the string
4134:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4135: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4136: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4137: ********************************** JvFileUtil;********************************
4138: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4139: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4140: procedure MoveFile(const FileName, DestName: TFileName);
4141: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4142: {$IFDEF COMPILER4_UP}
4143: function GetFileSize(const FileName: string): Int64;
4144: {$ELSE}
4145: function GetFileSize(const FileName: string): Longint;
4146: {$ENDIF}
4147: function FileDateTime(const FileName: string): TDateTime;
4148: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4149: function DeleteFiles(const FileMask: string): Boolean;
4150: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4151: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4152: function NormalDir(const DirName: string): string;
4153: function RemoveBackSlash(const DirName: string): string;
4154: function ValidFileName(const FileName: string): Boolean;
4155: function DirExists(Name: string): Boolean;
4156: procedure ForceDirectories(Dir: string);
4157: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4158:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4159: {$IFDEF COMPILER4_UP}
4160: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4161: {$ENDIF}
4162: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4163: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4164: {$IFDEF COMPILER4_UP}
4165: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4166: {$ENDIF}
4167: function GetTempDir: string;
4168: function GetWindowsDir: string;
4169: function GetSystemDir: string;
4170: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4171: {$IFDEF WIN32}
4172: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4173: function ShortToLongFileName(const ShortName: string): string;
4174: function ShortToLongPath(const ShortName: string): string;
4175: function LongToShortFileName(const LongName: string): string;
4176: function LongToShortPath(const LongName: string): string;
4177: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4178: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4179: {$ENDIF WIN32}
4180: {$IFNDEF COMPILER3_UP}
4181: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4182: {$ENDIF}
4183: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4184: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4185: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4186: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
```

```
4187:
4188: //*************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4189:  Procedure VariantClear( var V : Variant)
4190:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4191:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4192:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4193:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4194:  Procedure VariantSub( const src : Variant; var dst : Variant)
4195:  Procedure VariantMul( const src : Variant; var dst : Variant)
4196:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4197:  Procedure VariantMod( const src : Variant; var dst : Variant)
4198:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4199:  Procedure VariantOr( const src : Variant; var dst : Variant)
4200:  Procedure VariantXor( const src : Variant; var dst : Variant)
4201:  Procedure VariantShl( const src : Variant; var dst : Variant)
4202:  Procedure VariantShr( const src : Variant; var dst : Variant)
4203:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4204:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4205:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4206:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4207:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4208:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4209:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4210:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4211:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4212:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4213:  Function VariantNot( const V1 : Variant) : Variant
4214:  Function VariantNeg( const V1 : Variant) : Variant
4215:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4216:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4217:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4218:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4219:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4220:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4221:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4222:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4223:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4224:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4225: end;
4226:
4227: ************************************unit uPSI_JvgUtils;************************************
4228: function IsEven(I: Integer): Boolean;
4229: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4230: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4231: procedure SwapInt(var I1, I2: Integer);
4232: function Spaces(Count: Integer): string;
4233: function DupStr(const Str: string; Count: Integer): string;
4234: function DupChar(C: Char; Count: Integer): string;
4235: procedure Msg(const AMsg: string);
4236: function RectW(R: TRect): Integer;
4237: function RectH(R: TRect): Integer;
4238: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4239: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4240: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4241: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4242:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4243: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4244: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4245:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4246: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4247: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4248:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4249: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4250: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4251: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4252:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4253:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4254:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4255: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4256:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4257:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4258:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4259: procedure BringParentWindowToTop(Wnd: TWinControl);
4260: function GetParentForm(Control: TControl): TForm;
4261: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4262: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4263: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4264: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4265: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4266: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4267: function CalcMathString(AExpression: string): Single;
4268: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4269: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4270: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4271: procedure TypeStringOnKeyboard(const S: string);
4272: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4273: procedure DrawTextExtAligned(Canvas:TCanvas;const
      Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
```

```
4274: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4275: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4276: function ComponentToString(Component: TComponent): string;
4277: procedure StringToComponent(Component: TComponent; const Value: string);
4278: function PlayWaveResource(const ResName: string): Boolean;
4279: function UserName: string;
4280: function ComputerName: string;
4281: function CreateIniFileName: string;
4282: function ExpandString(const Str: string; Len: Integer): string;
4283: function Transliterate(const Str: string; RusToLat: Boolean): string;
4284: function IsSmallFonts: Boolean;
4285: function SystemColorDepth: Integer;
4286: function GetFileTypeJ(const FileName: string): TglFileType;
4287: Function GetFileType( hFile : THandle) : DWORD');
4288: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4289: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4290:
4291: { ************************************Utility routines of unit classes}
4292: function LineStart(Buffer, BufPos: PChar): PChar
4293: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4294:   'Strings: TStrings): Integer
4295:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4296:  Procedure RegisterClass( AClass : TPersistentClass)
4297:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4298:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4299:  Procedure UnRegisterClass( AClass : TPersistentClass)
4300:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4301:  Procedure UnRegisterModuleClasses( Module : HMODULE)
4302:  Function FindGlobalComponent( const Name : string) : TComponent
4303:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4304:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4305:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4306:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4307:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4308:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4309:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4310:  Procedure GlobalFixupReferences
4311:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4312:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4313:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4314:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4315:  Procedure RemoveFixups( Instance : TPersistent)
4316:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4317:  Procedure BeginGlobalLoading
4318:  Procedure NotifyGlobalLoading
4319:  Procedure EndGlobalLoading
4320:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4321:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4322:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4323:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4324:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
4325: // Function AllocateHWnd( Method : TWndMethod) : HWND
4326:  Procedure DeallocateHWnd( Wnd : HWND)
4327:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4328: ************************************unit uPSI_SqlTimSt and DB;*********************************
4329: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4330: Function VarSQLTimeStampCreate3: Variant;
4331: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4332: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4333: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4334: Function VarSQLTimeStamp : TVarType
4335: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4336: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4337: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4338: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4339: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4340: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4341: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4342: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4343: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4344: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4345: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4346: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4347: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4348:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4349: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4350: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4351: Procedure DisposeMem( var Buffer, Size : Integer)
4352: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4353: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4354: Function VarTypeToDataType( VarType : Integer) : TFieldType
4355: ********************************************unit JvStrings;*************************************
4356: {template functions}
4357: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4358: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4359: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4360: function RemoveMasterBlocks(const SourceStr: string): string;
4361: function RemoveFields(const SourceStr: string): string;
4362: {http functions}
```

```
4363: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4364: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4365: {set functions}
4366: procedure SplitSet(AText: string; AList: TStringList);
4367: function JoinSet(AList: TStringList): string;
4368: function FirstOfSet(const AText: string): string;
4369: function LastOfSet(const AText: string): string;
4370: function CountOfSet(const AText: string): Integer;
4371: function SetRotateRight(const AText: string): string;
4372: function SetRotateLeft(const AText: string): string;
4373: function SetPick(const AText: string; AIndex: Integer): string;
4374: function SetSort(const AText: string): string;
4375: function SetUnion(const Set1, Set2: string): string;
4376: function SetIntersect(const Set1, Set2: string): string;
4377: function SetExclude(const Set1, Set2: string): string;
4378: {replace any <,> etc by &lt; &gt;}
4379: function XMLSafe(const AText: string): string;
4380: {simple hash, Result can be used in Encrypt}
4381: function Hash(const AText: string): Integer;
4382: { Base64 encode and decode a string }
4383: function B64Encode(const S: AnsiString): AnsiString;
4384: function B64Decode(const S: AnsiString): AnsiString;
4385: {Basic encryption from a Borland Example}
4386: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4387: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4388: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4389: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4390: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4391: procedure CSVToTags(Src, Dst: TStringList);
4392: // converts a csv list to a tagged string list
4393: procedure TagsToCSV(Src, Dst: TStringList);
4394: // converts a tagged string list to a csv list
4395: // only fieldnames from the first record are scanned ib the other records
4396: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4397: {selects akey=avalue from Src and returns recordset in Dst}
4398: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4399: {filters Src for akey=avalue}
4400: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4401: {orders a tagged Src list by akey}
4402: function PosStr(const FindString, SourceString: string;
4403:   StartPos: Integer = 1): Integer;
4404: { PosStr searches the first occurrence of a substring FindString in a string
4405:   given by SourceString with case sensitivity (upper and lower case characters
4406:   are differed). This function returns the index value of the first character
4407:   of a specified substring from which it occurs in a given string starting with
4408:   StartPos character index. If a specified substring is not found Q_PosStr
4409:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4410: function PosStrLast(const FindString, SourceString: string): Integer;
4411: {finds the last occurance}
4412: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4413: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4414: { PosText searches the first occurrence of a substring FindString in a string
4415:   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
4416: function PosTextLast(const FindString, SourceString: string): Integer;
4417: {finds the last occurance}
4418: function NameValuesToXML(const AText: string): string;
4419: {$IFDEF MSWINDOWS}
4420: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4421: {$ENDIF MSWINDOWS}
4422: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4423: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4424: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4425: procedure SaveString(const AFile, AText: string);
4426: Procedure SaveStringasFile( const AFile, AText : string)
4427: function LoadStringJ(const AFile: string): string;
4428: Function LoadStringofFile( const AFile : string) : string
4429: Procedure SaveStringtoFile( const AFile, AText : string)
4430: Function LoadStringfromFile( const AFile : string) : string
4431: function HexToColor(const AText: string): TColor;
4432: function UppercaseHTMLTags(const AText: string): string;
4433: function LowercaseHTMLTags(const AText: string): string;
4434: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4435: function RelativePath(const ASrc, ADst: string): string;
4436: function GetToken(var Start: Integer; const SourceText: string): string;
4437: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4438: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4439: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4440: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4441: // parses the beginning of an attribute: space + alpha character
4442: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4443: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4444: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4445: // parses all name=value attributes to the attributes TStringList
4446: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4447: // checks if a name="value" pair exists and returns any value
4448: function GetStrValue(const AText, AName, ADefault: string): string;
4449: // retrieves string value from a line like:
```

```
4450: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4451: // returns ADefault when not found
4452: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4453: // same for a color
4454: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4455: // same for an Integer
4456: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4457: // same for a float
4458: function GetBoolValue(const AText, AName: string): Boolean;
4459: // same for Boolean but without default
4460: function GetValue(const AText, AName: string): string;
4461: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4462: procedure SetValue(var AText: string; const AName, AValue: string);
4463: // sets a string value in a line
4464: procedure DeleteValue(var AText: string; const AName: string);
4465: // deletes a AName="value" pair from AText
4466: procedure GetNames(AText: string; AList: TStringList);
4467: // get a list of names from a string with name="value" pairs
4468: function GetHTMLColor(AColor: TColor): string;
4469: // converts a color value to the HTML hex value
4470: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4471: // finds a string backward case sensitive
4472: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4473: // finds a string backward case insensitive
4474: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4475:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4476: // finds a text range, e.g. <TD>....</TD> case sensitive
4477: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4478:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4479: // finds a text range, e.g. <TD>....</td> case insensitive
4480: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4481:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4482: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4483: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4484:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4485: // finds a text range backward, e.g. <TD>....</td> case insensitive
4486: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4487:   var RangeEnd: Integer): Boolean;
4488: // finds a HTML or XML tag:  <....>
4489: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4490:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4491: // finds the innertext between opening and closing tags
4492: function Easter(NYear: Integer): TDateTime;
4493: // returns the easter date of a year.
4494: function GetWeekNumber(Today: TDateTime): string;
4495: //gets a datecode. Returns year and weeknumber in format: YYWW
4496: function ParseNumber(const S: string): Integer;
4497: // parse number returns the last position, starting from 1
4498: function ParseDate(const S: string): Integer;
4499: // parse a SQL style data string from positions 1,
4500: // starts and ends with #
4501:
4502: ****************************************unit JvJCLUtils;****************************************
4503:
4504: function VarIsInt(Value: Variant): Boolean;
4505:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4506: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4507: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4508: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4509: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4510: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4511: function GetWordOnPos(const S: string; const P: Integer): string;
4512: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4513: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4514: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4515: { GetWordOnPosEx working like GetWordOnPos function, but
4516:   also returns Word position in iBeg, iEnd variables }
4517: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4518: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4519: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4520: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4521: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4522: { GetEndPosCaret returns the caret position of the last char. For the position
4523:   after the last char of Text you must add 1 to the returned X value. }
4524: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
4525: { GetEndPosCaret returns the caret position of the last char. For the position
4526:   after the last char of Text you must add 1 to the returned X value. }
4527: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4528: function SubStrBySeparator(const S:string;const Index:Integer;const
     Separator:string;StartIndex:Int=1):string;
4529: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
     Separator:WideString;StartIndex:Int:WideString;
4530: { SubStrEnd same to previous function but Index numerated from the end of string }
4531: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4532: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4533: function SubWord(P: PChar; var P2: PChar): string;
4534: function CurrencyByWord(Value: Currency): string;
4535: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4536: function GetLineByPos(const S: string; const Pos: Integer): Integer;
```

```
4537: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4538: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4539: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4540: { ReplaceString searches for all substrings, OldPattern,
4541:   in a string, S, and replaces them with NewPattern }
4542: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4543: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
      WideString;StartIndex:Integer=1):WideString;
4544: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4545: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4546: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4547: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4548:
4549: { Next 4 function for russian chars transliterating.
4550:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4551: procedure Dos2Win(var S: AnsiString);
4552: procedure Win2Dos(var S: AnsiString);
4553: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4554: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4555: function Win2Koi(const S: AnsiString): AnsiString;
4556: { FillString fills the string Buffer with Count Chars }
4557: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4558: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4559: { MoveString copies Count Chars from Source to Dest }
4560: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE} overload;
4561: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4562:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4563: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4564: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4565: { MoveWideChar copies Count WideChars from Source to Dest }
4566: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
      SUPPORTS_INLINE}
4567: { FillNativeChar fills Buffer with Count NativeChars }
4568: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4569: { MoveWideChar copies Count WideChars from Source to Dest }
4570: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4571: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4572: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4573: { Spaces returns string consists on N space chars }
4574: function Spaces(const N: Integer): string;
4575: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4576: function AddSpaces(const S: string; const N: Integer): string;
4577: function SpacesW(const N: Integer): WideString;
4578: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4579: { function LastDateRUS for russian users only }
4580: { returns date relative to current date: 'ääà äíý íàçàä' }
4581: function LastDateRUS(const Dat: TDateTime): string;
4582: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4583: function CurrencyToStr(const Cur: Currency): string;
4584: { HasChar returns True, if Char, Ch, contains in string, S }
4585: function HasChar(const Ch: Char; const S: string): Boolean;
4586: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4587: function HasAnyChar(const Chars: string; const S: string): Boolean;
4588: {$IFNDEF COMPILER12_UP}
4589: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4590: {$ENDIF ~COMPILER12_UP}
4591: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
      SUPPORTS_INLINE}
4592: function CountOfChar(const Ch: Char; const S: string): Integer;
4593: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4594: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4595: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4596: function StrPosW(S, SubStr: PWideChar): PWideChar;
4597: function StrLenW(S: PWideChar): Integer;
4598: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4599: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4600: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4601: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4602: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4603:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4604: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4605: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4606: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4607: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4608: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
4609: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4610: Function ScreenPixelFormat : TPixelFormat
4611: Function ScreenColorCount : Integer
4612: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4613: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4614: //  SIRegister_TJvGradient(CL);
4615:
```

```
4616: {************************************ files routines}
4617: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4618: const
4619:   {$IFDEF MSWINDOWS}
4620:   DefaultCaseSensitivity = False;
4621:   {$ENDIF MSWINDOWS}
4622:   {$IFDEF UNIX}
4623:   DefaultCaseSensitivity = True;
4624:   {$ENDIF UNIX}
4625: { GenTempFileName returns temporary file name on
4626:   drive, there FileName is placed }
4627: function GenTempFileName(FileName: string): string;
4628: { GenTempFileNameExt same to previous function, but
4629:   returning filename has given extension, FileExt }
4630: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4631: { ClearDir clears folder Dir }
4632: function ClearDir(const Dir: string): Boolean;
4633: { DeleteDir clears and than delete folder Dir }
4634: function DeleteDir(const Dir: string): Boolean;
4635: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4636: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4637: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4638:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4639: function FileEquMasks(FileName, Masks: TFileName;
4640:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4641: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4642: {$IFDEF MSWINDOWS}
4643: { LZFileExpand expand file, FileSource,
4644:   into FileDest. Given file must be compressed, using MS Compress program }
4645: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4646: {$ENDIF MSWINDOWS}
4647: { FileGetInfo fills SearchRec record for specified file attributes}
4648: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4649: { HasSubFolder returns True, if folder APath contains other folders }
4650: function HasSubFolder(APath: TFileName): Boolean;
4651: { IsEmptyFolder returns True, if there are no files or
4652:   folders in given folder, APath}
4653: function IsEmptyFolder(APath: TFileName): Boolean;
4654: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4655: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4656: { AddPath returns FileName with Path, if FileName not contain any path }
4657: function AddPath(const FileName, Path: TFileName): TFileName;
4658: function AddPaths(const PathList, Path: string): string;
4659: function ParentPath(const Path: TFileName): TFileName;
4660: function FindInPath(const FileName, PathList: string): TFileName;
4661: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4662: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4663: { HasParam returns True, if program running with specified parameter, Param }
4664: function HasParam(const Param: string): Boolean;
4665: function HasSwitch(const Param: string): Boolean;
4666: function Switch(const Param: string): string;
4667: { ExePath returns ExtractFilePath(ParamStr(0)) }
4668: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4669: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4670: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4671: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4672: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4673: {**** Graphic routines }
4674: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4675: function IsTTFontSelected(const DC: HDC): Boolean;
4676: function KeyPressed(VK: Integer): Boolean;
4677: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4678: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4679: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4680: {**** Color routines }
4681: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4682: function RGBToBGR(Value: Cardinal): Cardinal;
4683: //function ColorToPrettyName(Value: TColor): string;
4684: //function PrettyNameToColor(const Value: string): TColor;
4685: {**** other routines }
4686: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4687: function IntPower(Base, Exponent: Integer): Integer;
4688: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4689: function StrToBool(const S: string): Boolean;
4690: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4691: function VarToInt(V: Variant): Integer;
4692: function VarToFloat(V: Variant): Double;
4693: { following functions are not documented because they not work properly sometimes, so do not use them }
4694: // (rom) ReplaceStrings1, GetSubStr removed
4695: function GetLongFileName(const FileName: string): string;
4696: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4697: function GetParameter: string;
4698: function GetComputerID: string;
4699: function GetComputerName: string;
4700: {**** string routines }
4701: { ReplaceAllStrings searches for all substrings, Words,
4702:   in a string, S, and replaces them with Frases with the same Index. }
4703: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4704: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
```

```
4705:   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 }
4706: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4707: { CountOfLines calculates the lines count in a string, S,
4708:   each line must be separated from another with CrLf sequence }
4709: function CountOfLines(const S: string): Integer;
4710: { DeleteLines deletes all lines from strings which in the words,  words.
4711:   The word of will be deleted from strings. }
4712: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4713: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4714:   Lines contained only spaces also deletes. }
4715: procedure DeleteEmptyLines(Ss: TStrings);
4716: { SQLAddWhere addes or modifies existing where-statement, where,
4717:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4718:   it must be started on the begining of any line }
4719: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4720: {**** files routines - }
4721: {$IFDEF MSWINDOWS}
4722: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4723:   Resource can be compressed using MS Compress program}
4724: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4725: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
      Boolean;
4726: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4727: {$ENDIF MSWINDOWS}
4728: { IniReadSection read section, Section, from ini-file,
4729:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4730:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4731: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4732: { LoadTextFile load text file, FileName, into string }
4733: function LoadTextFile(const FileName: TFileName): string;
4734: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4735: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4736: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4737: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4738: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4739: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4740: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4741: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4742: { RATextCalcHeight calculate needed height for
4743:   correct output, using RATextOut or RATextOutEx functions }
4744: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4745: { Cinema draws some visual effect }
4746: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4747: { Roughed fills rect with special 3D pattern }
4748: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4749: { 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 }
4750: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4751: { TextWidth calculate text with for writing using standard desktop font }
4752: function TextWidth(const AStr: string): Integer;
4753: { TextHeight calculate text height for writing using standard desktop font }
4754: function TextHeight(const AStr: string): Integer;
4755: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4756: procedure Error(const Msg: string);
4757: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4758:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4759: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4760: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4761:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4762: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4763:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4764: function ItemHtPlain(const Text: string): string;
4765: { ClearList - clears list of TObject }
4766: procedure ClearList(List: TList);
4767: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4768: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4769: { RTTI support }
4770: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4771: function GetPropStr(Obj: TObject; const PropName: string): string;
4772: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4773: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4774: procedure PrepareIniSection(Ss: TStrings);
4775: { following functions are not documented because they are don't work properly, so don't use them }
4776: // (rom) from JvBandWindows to make it obsolete
4777: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4778: // (rom) from JvBandUtils to make it obsolete
4779: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4780: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4781: function CreateIconFromClipboard: TIcon;
4782: { begin JvIconClipboardUtils } { Icon clipboard routines }
4783: function CF_ICON: Word;
4784: procedure AssignClipboardIcon(Icon: TIcon);
4785: { Real-size icons support routines (32-bit only) }
4786: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4787: function CreateRealSizeIcon(Icon: TIcon): HICON;
4788: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4789: {end JvIconClipboardUtils }
4790: function CreateScreenCompatibleDC: HDC;
```

```
4791: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF}
4792: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4793: { begin JvRLE } // (rom) changed API for inclusion in JCL
4794: procedure RleCompressTo(InStream, OutStream: TStream);
4795: procedure RleDecompressTo(InStream, OutStream: TStream);
4796: procedure RleCompress(Stream: TStream);
4797: procedure RleDecompress(Stream: TStream);
4798: { end JvRLE } { begin JvDateUtil }
4799: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4800: function IsLeapYear(AYear: Integer): Boolean;
4801: function DaysInAMonth(const AYear, AMonth: Word): Word;
4802: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4803: function FirstDayOfPrevMonth: TDateTime;
4804: function LastDayOfPrevMonth: TDateTime;
4805: function FirstDayOfNextMonth: TDateTime;
4806: function ExtractDay(ADate: TDateTime): Word;
4807: function ExtractMonth(ADate: TDateTime): Word;
4808: function ExtractYear(ADate: TDateTime): Word;
4809: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4810: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4811: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4812: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4813: function ValidDate(ADate: TDateTime): Boolean;
4814: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4815: function MonthsBetween(Date1, Date2: TDateTime): Double;
4816: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4817: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4818: function DaysBetween(Date1, Date2: TDateTime): Longint;
4819: { The same as previous but if Date2 < Date1 result = 0 }
4820: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4821: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4822: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4823: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4824: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4825: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4826: { String to date conversions }
4827: function GetDateOrder(const DateFormat: string): TDateOrder;
4828: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4829: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4830: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4831: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4832: //function DefDateFormat(AFourDigitYear: Boolean): string;
4833: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4834: function FormatLongDate(Value: TDateTime): string;
4835: function FormatLongDateTime(Value: TDateTime): string;
4836: { end JvDateUtil }
4837: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4838: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4839: { begin JvStrUtils } { ** Common string handling routines ** }
4840: {$IFDEF UNIX}
4841: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4842:   const ToCode, FromCode: AnsiString): Boolean;
4843: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4844: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4845: function OemStrToAnsi(const S: AnsiString): AnsiString;
4846: function AnsiStrToOem(const S: AnsiString): AnsiString;
4847: {$ENDIF UNIX}
4848: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4849: { StrToOem translates a string from the Windows character set into the OEM character set. }
4850: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4851: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4852: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4853: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4854: function ReplaceStr(const S, Srch, Replace: string): string;
4855: { Returns string with every occurrence of Srch string replaced with Replace string. }
4856: function DelSpace(const S: string): string;
4857: { DelSpace return a string with all white spaces removed. }
4858: function DelChars(const S: string; Chr: Char): string;
4859: { DelChars return a string with all Chr characters removed. }
4860: function DelBSpace(const S: string): string;
4861: { DelBSpace trims leading spaces from the given string. }
4862: function DelESpace(const S: string): string;
4863: { DelESpace trims trailing spaces from the given string. }
4864: function DelRSpace(const S: string): string;
4865: { DelRSpace trims leading and trailing spaces from the given string. }
4866: function DelSpace1(const S: string): string;
4867: { DelSpace1 return a string with all non-single white spaces removed. }
4868: function Tab2Space(const S: string; Numb: Byte): string;
4869: { Tab2Space converts any tabulation character in the given string to the
4870:   Numb spaces characters. }
4871: function NPos(const C: string; S: string; N: Integer): Integer;
4872: { NPos searches for a N-th position of substring C in a given string. }
4873: function MakeStr(C: Char; N: Integer): string; overload;
4874: {$IFNDEF COMPILER12_UP}
4875: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4876: {$ENDIF !COMPILER12_UP}
4877: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4878: { MakeStr return a string of length N filled with character C. }
```

```
4879: function AddChar(C: Char; const S: string; N: Integer): string;
4880: { AddChar return a string left-padded to length N with characters C. }
4881: function AddCharR(C: Char; const S: string; N: Integer): string;
4882: { AddCharR return a string right-padded to length N with characters C. }
4883: function LeftStr(const S: string; N: Integer): string;
4884: { LeftStr return a string right-padded to length N with blanks. }
4885: function RightStr(const S: string; N: Integer): string;
4886: { RightStr return a string left-padded to length N with blanks. }
4887: function CenterStr(const S: string; Len: Integer): string;
4888: { CenterStr centers the characters in the string based upon the Len specified. }
4889: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4890: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4891:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4892: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4893: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4894: function Copy2Symb(const S: string; Symb: Char): string;
4895: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4896: function Copy2SymbDel(var S: string; Symb: Char): string;
4897: { Copy2SymbDel returns a substring of a string S from begining to first
4898:   character Symb and removes this substring from S. }
4899: function Copy2Space(const S: string): string;
4900: { Copy2Symb returns a substring of a string S from begining to first white space. }
4901: function Copy2SpaceDel(var S: string): string;
4902: { Copy2SpaceDel returns a substring of a string S from begining to first
4903:   white space and removes this substring from S. }
4904: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4905: { Returns string, with the first letter of each word in uppercase,
4906:   all other letters in lowercase. Words are delimited by WordDelims. }
4907: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4908: { WordCount given a set of word delimiters, returns number of words in S. }
4909: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4910: { Given a set of word delimiters, returns start position of N'th word in S. }
4911: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4912: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4913: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4914: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4915:   delimiters, return the N'th word in S. }
4916: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4917: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4918:   that started from position Pos. }
4919: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4920: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4921: function QuotedString(const S: string; Quote: Char): string;
4922: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4923: function ExtractQuotedString(const S: string; Quote: Char): string;
4924: { ExtractQuotedString removes the Quote characters from the beginning and
4925:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4926: function FindPart(const HelpWilds, InputStr: string): Integer;
4927: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4928: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4929: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4930: function XorString(const Key, Src: ShortString): ShortString;
4931: function XorEncode(const Key, Source: string): string;
4932: function XorDecode(const Key, Source: string): string;
4933: { ** Command line routines ** }
4934: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4935: { ** Numeric string handling routines ** }
4936: function Numb2USA(const S: string): string;
4937: { Numb2USA converts numeric string S to USA-format. }
4938: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4939: { Dec2Hex converts the given value to a hexadecimal string representation
4940:   with the minimum number of digits (A) specified. }
4941: function Hex2Dec(const S: string): Longint;
4942: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4943: function Dec2Numb(N: Int64; A, B: Byte): string;
4944: { Dec2Numb converts the given value to a string representation with the
4945:   base equal to B and with the minimum number of digits (A) specified. }
4946: function Numb2Dec(S: string; B: Byte): Int64;
4947: { Numb2Dec converts the given B-based numeric string to the corresponding
4948:   integer value. }
4949: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4950: { IntToBin converts the given value to a binary string representation
4951:   with the minimum number of digits specified. }
4952: function IntToRoman(Value: Longint): string;
4953: { IntToRoman converts the given value to a roman numeric string representation. }
4954: function RomanToInt(const S: string): Longint;
4955: { RomanToInt converts the given string to an integer value. If the string
4956:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4957: function FindNotBlankCharPos(const S: string): Integer;
4958: function FindNotBlankCharPosW(const S: WideString): Integer;
4959: function AnsiChangeCase(const S: string): string;
4960: function WideChangeCase(const S: string): string;
4961: function StartsText(const SubStr, S: string): Boolean;
4962: function EndsText(const SubStr, S: string): Boolean;
4963: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4964: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4965: {end JvStrUtils}
4966: {$IFDEF UNIX}
4967: function GetTempFileName(const Prefix: AnsiString): AnsiString;
```

```
4968: {$ENDIF UNIX}
4969: { begin JvFileUtil }
4970: function FileDateTime(const FileName: string): TDateTime;
4971: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4972: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4973: function NormalDir(const DirName: string): string;
4974: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4975: function ValidFileName(const FileName: string): Boolean;
4976: {$IFDEF MSWINDOWS}
4977: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4978: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4979: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4980: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4981: {$ENDIF MSWINDOWS}
4982: function GetWindowsDir: string;
4983: function GetSystemDir: string;
4984: function ShortToLongFileName(const ShortName: string): string;
4985: function LongToShortFileName(const LongName: string): string;
4986: function ShortToLongPath(const ShortName: string): string;
4987: function LongToShortPath(const LongName: string): string;
4988: {$IFDEF MSWINDOWS}
4989: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4990: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4991: {$ENDIF MSWINDOWS}
4992: { end JvFileUtil }
4993: // Works like PtInRect but includes all edges in comparision
4994: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4995: // Works like PtInRect but excludes all edges from comparision
4996: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4997: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4998: function IsFourDigitYear: Boolean;
4999: { moved from JvJVCLUTils }
5000: //Open an object with the shell (url or something like that)
5001: function OpenObject(const Value: string): Boolean; overload;
5002: function OpenObject(Value: PChar): Boolean; overload;
5003: {$IFDEF MSWINDOWS}
5004: //Raise the last Exception
5005: procedure RaiseLastWin32; overload;
5006: procedure RaiseLastWin32(const Text: string); overload;
5007: //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
5008: function GetFileVersion(const AFileName: string): Cardinal;
5009: {$EXTERNALSYM GetFileVersion}
5010: //Get version of Shell.dll
5011: function GetShellVersion: Cardinal;
5012: {$EXTERNALSYM GetShellVersion}
5013: // CD functions  on HW
5014: procedure OpenCdDrive;
5015: procedure CloseCdDrive;
5016: // returns True if Drive is accessible
5017: function DiskInDrive(Drive: Char): Boolean;
5018: {$ENDIF MSWINDOWS}
5019: //Same as linux function ;)
5020: procedure PError(const Text: string);
5021: // execute a program without waiting
5022: procedure Exec(const FileName, Parameters, Directory: string);
5023: // execute a program and wait for it to finish
5024: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5025: // returns True if this is the first instance of the program that is running
5026: function FirstInstance(const ATitle: string): Boolean;
5027: // restores a window based on it's classname and Caption. Either can be left empty
5028: // to widen the search
5029: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5030: // manipulate the traybar and start button
5031: procedure HideTraybar;
5032: procedure ShowTraybar;
5033: procedure ShowStartButton(Visible: Boolean = True);
5034: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5035: procedure MonitorOn;
5036: procedure MonitorOff;
5037: procedure LowPower;
5038: // send a key to the window named AppName
5039: function SendKey(const AppName: string; Key: Char): Boolean;
5040: {$IFDEF MSWINDOWS}
5041: // returns a list of all win currently visible, the Objects property is filled with their window handle
5042: procedure GetVisibleWindows(List: TStrings);
5043: Function GetVisibleWindowsF( List : TStrings):TStrings');
5044: // associates an extension to a specific program
5045: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5046: procedure AddToRecentDocs(const FileName: string);
5047: function GetRecentDocs: TStringList;
5048: {$ENDIF MSWINDOWS}
5049: function CharIsMoney(const Ch: Char): Boolean;
5050: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5051: function IntToExtended(I: Integer): Extended;
5052: { GetChangedText works out the new text given the current cursor pos & the key pressed
5053:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5054: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
```

```
5055: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5056: //function StrIsInteger(const S: string): Boolean;
5057: function StrIsFloatMoney(const Ps: string): Boolean;
5058: function StrIsDateTime(const Ps: string): Boolean;
5059: function PreformatDateString(Ps: string): string;
5060: function BooleanToInteger(const B: Boolean): Integer;
5061: function StringToBoolean(const Ps: string): Boolean;
5062: function SafeStrToDateTime(const Ps: string): TDateTime;
5063: function SafeStrToDate(const Ps: string): TDateTime;
5064: function SafeStrToTime(const Ps: string): TDateTime;
5065: function StrDelete(const psSub, psMain: string): string;
5066:   { returns the fractional value of pcValue}
5067: function TimeOnly(pcValue: TDateTime): TTime;
5068: { returns the integral value of pcValue }
5069: function DateOnly(pcValue: TDateTime): TDate;
5070: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5071: const { TDateTime value used to signify Null value}
5072:   NullEquivalentDate: TDateTime = 0.0;
5073: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5074: // Replacement for Win32Check to avoid platform specific warnings in D6
5075: function OSCheck(RetVal: Boolean): Boolean;
5076: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5077:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5078:   not be forced to use FileCtrl unnecessarily }
5079: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5080: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5081: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5082: function MinimizeString(const S: string; const MaxLen: Integer): string;
5083: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
     SW_SHOWDEFAULT);
5084: { 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.}
5085: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5086: {$ENDIF MSWINDOWS}
5087: procedure ResourceNotFound(ResID: PChar);
5088: function EmptyRect: TRect;
5089: function RectWidth(R: TRect): Integer;
5090: function RectHeight(R: TRect): Integer;
5091: function CompareRect(const R1, R2: TRect): Boolean;
5092: procedure RectNormalize(var R: TRect);
5093: function RectIsSquare(const R: TRect): Boolean;
5094: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5095: //If AMaxSize = -1 ,then auto calc Square's max size
5096: {$IFDEF MSWINDOWS}
5097: procedure FreeUnusedOle;
5098: function GetWindowsVersion: string;
5099: function LoadDLL(const LibName: string): THandle;
5100: function RegisterServer(const ModuleName: string): Boolean;
5101: function UnregisterServer(const ModuleName: string): Boolean;
5102: {$ENDIF MSWINDOWS}
5103: { String routines }
5104: function GetEnvVar(const VarName: string): string;
5105: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5106: function StringToPChar(var S: string): PChar;
5107: function StrPAlloc(const S: string): PChar;
5108: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5109: function DropT(const S: string): string;
5110: { Memory routines }
5111: function AllocMemo(Size: Longint): Pointer;
5112: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5113: procedure FreeMemo(var fpBlock: Pointer);
5114: function GetMemoSize(fpBlock: Pointer): Longint;
5115: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5116: { Manipulate huge pointers routines }
5117: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5118: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5119: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5120: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5121: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5122: function WindowClassName(Wnd: THandle): string;
5123: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5124: procedure ActivateWindow(Wnd: THandle);
5125: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5126: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5127: { SetWindowTop put window to top without recreating window }
5128: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5129: procedure CenterWindow(Wnd: THandle);
5130: function MakeVariant(const Values: array of Variant): Variant;
5131: { Convert dialog units to pixels and backwards }
5132: {$IFDEF MSWINDOWS}
5133: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5134: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5135: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5136: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5137: {$ENDIF MSWINDOWS}
5138: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5139: {$IFDEF BCB}
5140: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
```

```
5141: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5142: {$ELSE}
5143: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5144: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5145: {$ENDIF BCB}
5146: {$IFDEF MSWINDOWS}
5147: { BrowseForFolderNative displays Browse For Folder dialog }
5148: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5149: {$ENDIF MSWINDOWS}
5150: procedure AntiAlias(Clip: TBitmap);
5151: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5152: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5153:   ABitmap: TBitmap; const SourceRect: TRect);
5154: function IsTrueType(const FontName: string): Boolean;
5155: // Removes all non-numeric characters from AValue and returns the resulting string
5156: function TextToValText(const AValue: string): string;
5157: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5158: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5159: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5160: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5161: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5162:
5163: ******************************************************unit uPSI_JvTFUtils;
5164:  Function JExtractYear( ADate : TDateTime) : Word
5165:  Function JExtractMonth( ADate : TDateTime) : Word
5166:  Function JExtractDay( ADate : TDateTime) : Word
5167:  Function ExtractHours( ATime : TDateTime) : Word
5168:  Function ExtractMins( ATime : TDateTime) : Word
5169:  Function ExtractSecs( ATime : TDateTime) : Word
5170:  Function ExtractMSecs( ATime : TDateTime) : Word
5171:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5172:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5173:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5174:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5175:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5176:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5177:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5178:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5179:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5180:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5181:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5182:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5183:  Procedure EnsureMonth( Month : Word)
5184:  Procedure EnsureDOW( DOW : Word)
5185:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5186:  Function Lesser( N1, N2 : Integer) : Integer
5187:  Function Greater( N1, N2 : Integer) : Integer
5188:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5189:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5190:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5191:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5192:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5193:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5194:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
     AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5195:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
     HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5196:  Function JRectWidth( ARect : TRect) : Integer
5197:  Function JRectHeight( ARect : TRect) : Integer
5198:  Function JEmptyRect : TRect
5199:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5200:
5201: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5202: begin
5203:  Procedure HideTaskBarButton( hWindow : HWND)
5204:  Function msLoadStr( ID : Integer) : String
5205:  Function msFormat( fmt : String; params : array of const) : String
5206:  Function msFileExists( const FileName : String) : Boolean
5207:  Function msIntToStr( Int : Int64) : String
5208:  Function msStrPas( const Str : PChar) : String
5209:  Function msRenameFile( const OldName, NewName : String) : Boolean
5210:  Function CutFileName( s : String) : String
5211:  Function GetVersionInfo( var VersionString : String) : DWORD
5212:  Function FormatTime( t : Cardinal) : String
5213:  Function msCreateDir( const Dir : string) : Boolean
5214:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5215:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5216:  Function msStrLen( Str : PChar) : Integer
5217:  Function msDirectoryExists( const Directory : String) : Boolean
5218:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5219:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5220:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5221:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5222:  Function GetTextFromFile( Filename : String) : string
5223:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5224:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5225:  Function msStrToInt( s : String) : Integer
5226:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
```

```
5227: end;
5228:
5229: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5230: begin
5231:   //TDynFloatArray', 'array of Extended
5232:   TDynLWordArray', 'array of LongWord
5233:   TDynLIntArray', 'array of LongInt
5234:   TDynFloatMatrix', 'array of TDynFloatArray
5235:   TDynLWordMatrix', 'array of TDynLWordArray
5236:   TDynLIntMatrix', 'array of TDynLIntArray
5237:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5238:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5239:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5240:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5241:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5242:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5243:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5244:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5245:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5246:  Function MNorm( const X : TDynFloatArray) : Extended
5247:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5248:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5249:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5250:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5251:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5252:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5253:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5254:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5255:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5256:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5257:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5258:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5259:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5260: end;
5261:
5262: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5263: begin
5264:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5265:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5266:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5267:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5268:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5269:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5270:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5271:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5272:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5273:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5274:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5275:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5276:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5277:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5278:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5279:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5280:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5281:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5282:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5283:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5284:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5285:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5286:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5287:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5288:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5289:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5290:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5291:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5292:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5293:  'ESBLn10','Extended').setExtended( 2.3025850929940456802);
5294:  'ESBLnPi','Extended').setExtended( 1.1447298858494001741);
5295:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5296:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5297:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5298:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5299:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5300:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5301:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5302:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5303:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5304:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5305:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5306:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5307:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5308:  'ESBPiOn4','Extended').setExtended( 0.78539816339744483096);
5309:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5310:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5311:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5312:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5313:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5314:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5315:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
```

```
5316:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5317:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5318:   //LongWord', 'Cardinal
5319:   TBitList', 'Word
5320:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5321:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5322:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5323:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5324:  Function SameFloat( const X1, X2 : Extended) : Boolean
5325:  Function FloatIsZero( const X : Extended) : Boolean
5326:  Function FloatIsPositive( const X : Extended) : Boolean
5327:  Function FloatIsNegative( const X : Extended) : Boolean
5328:  Procedure IncLim( var B : Byte; const Limit : Byte)
5329:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5330:  Procedure IncLimW( var B : Word; const Limit : Word)
5331:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5332:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5333:  Procedure DecLim( var B : Byte; const Limit : Byte)
5334:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5335:  Procedure DecLimW( var B : Word; const Limit : Word)
5336:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5337:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5338:  Function MaxB( const B1, B2 : Byte) : Byte
5339:  Function MinB( const B1, B2 : Byte) : Byte
5340:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5341:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5342:  Function MaxW( const B1, B2 : Word) : Word
5343:  Function MinW( const B1, B2 : Word) : Word
5344:  Function esbMaxI( const B1, B2 : Integer) : Integer
5345:  Function esbMinI( const B1, B2 : Integer) : Integer
5346:  Function MaxL( const B1, B2 : LongInt) : LongInt
5347:  Function MinL( const B1, B2 : LongInt) : LongInt
5348:  Procedure SwapB( var B1, B2 : Byte)
5349:  Procedure SwapSI( var B1, B2 : ShortInt)
5350:  Procedure SwapW( var B1, B2 : Word)
5351:  Procedure SwapI( var B1, B2 : SmallInt)
5352:  Procedure SwapL( var B1, B2 : LongInt)
5353:  Procedure SwapI32( var B1, B2 : Integer)
5354:  Procedure SwapC( var B1, B2 : LongWord)
5355:  Procedure SwapInt64( var X, Y : Int64)
5356:  Function esbSign( const B : LongInt) : ShortInt
5357:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5358:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5359:  Function Max3Word( const X1, X2, X3 : Word) : Word
5360:  Function Min3Word( const X1, X2, X3 : Word) : Word
5361:  Function MaxBArray( const B : array of Byte) : Byte
5362:  Function MaxWArray( const B : array of Word) : Word
5363:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5364:  Function MaxIArray( const B : array of Integer) : Integer
5365:  Function MaxLArray( const B : array of LongInt) : LongInt
5366:  Function MinBArray( const B : array of Byte) : Byte
5367:  Function MinWArray( const B : array of Word) : Word
5368:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5369:  Function MinIArray( const B : array of Integer) : Integer
5370:  Function MinLArray( const B : array of LongInt) : LongInt
5371:  Function SumBArray( const B : array of Byte) : Byte
5372:  Function SumBArray2( const B : array of Byte) : Word
5373:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5374:  Function SumSIArray2( const B : array of ShortInt) : Integer
5375:  Function SumWArray( const B : array of Word) : Word
5376:  Function SumWArray2( const B : array of Word) : LongInt
5377:  Function SumIArray( const B : array of Integer) : Integer
5378:  Function SumLArray( const B : array of LongInt) : LongInt
5379:  Function SumLWArray( const B : array of LongWord) : LongWord
5380:  Function ESBDigits( const X : LongWord) : Byte
5381:  Function BitsHighest( const X : LongWord) : Integer
5382:  Function ESBBitsNeeded( const X : LongWord) : Integer
5383:  Function esbGCD( const X, Y : LongWord) : LongWord
5384:  Function esbLCM( const X, Y : LongInt) : Int64
5385:  //Function esbLCM( const X, Y : LongInt) : LongInt
5386:  Function RelativePrime( const X, Y : LongWord) : Boolean
5387:  Function Get87ControlWord : TBitList
5388:  Procedure Set87ControlWord( const CWord : TBitList)
5389:  Procedure SwapExt( var X, Y : Extended)
5390:  Procedure SwapDbl( var X, Y : Double)
5391:  Procedure SwapSing( var X, Y : Single)
5392:  Function esbSgn( const X : Extended) : ShortInt
5393:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5394:  Function ExtMod( const X, Y : Extended) : Extended
5395:  Function ExtRem( const X, Y : Extended) : Extended
5396:  Function CompMOD( const X, Y : Comp) : Comp
5397:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5398:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5399:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5400:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5401:  Function MaxExt( const X, Y : Extended) : Extended
5402:  Function MinExt( const X, Y : Extended) : Extended
5403:  Function MaxEArray( const B : array of Extended) : Extended
5404:  Function MinEArray( const B : array of Extended) : Extended
```

```
5405:  Function MaxSArray( const B : array of Single) : Single
5406:  Function MinSArray( const B : array of Single) : Single
5407:  Function MaxCArray( const B : array of Comp) : Comp
5408:  Function MinCArray( const B : array of Comp) : Comp
5409:  Function SumSArray( const B : array of Single) : Single
5410:  Function SumEArray( const B : array of Extended) : Extended
5411:  Function SumSqEArray( const B : array of Extended) : Extended
5412:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5413:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5414:  Function SumCArray( const B : array of Comp) : Comp
5415:  Function FactorialX( A : LongWord) : Extended
5416:  Function PermutationX( N, R : LongWord) : Extended
5417:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5418:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5419:  Function esbGeometricMean( const X : array of Extended) : Extended
5420:  Function esbHarmonicMean( const X : array of Extended) : Extended
5421:  Function ESBMean( const X : array of Extended) : Extended
5422:  Function esbSampleVariance( const X : array of Extended) : Extended
5423:  Function esbPopulationVariance( const X : array of Extended) : Extended
5424:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5425:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5426:  Function GetMedian( const SortedX : array of Extended) : Extended
5427:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5428:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5429:  Function ESBMagnitude( const X : Extended) : Integer
5430:  Function ESBTan( Angle : Extended) : Extended
5431:  Function ESBCot( Angle : Extended) : Extended
5432:  Function ESBCosec( const Angle : Extended) : Extended
5433:  Function ESBSec( const Angle : Extended) : Extended
5434:  Function ESBArcTan( X, Y : Extended) : Extended
5435:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5436:  Function ESBArcCos( const X : Extended) : Extended
5437:  Function ESBArcSin( const X : Extended) : Extended
5438:  Function ESBArcSec( const X : Extended) : Extended
5439:  Function ESBArcCosec( const X : Extended) : Extended
5440:  Function ESBLog10( const X : Extended) : Extended
5441:  Function ESBLog2( const X : Extended) : Extended
5442:  Function ESBLogBase( const X, Base : Extended) : Extended
5443:  Function Pow2( const X : Extended) : Extended
5444:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5445:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5446:  Function XtoY( const X, Y : Extended) : Extended
5447:  Function esbTenToY( const Y : Extended) : Extended
5448:  Function esbTwoToY( const Y : Extended) : Extended
5449:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5450:  Function esbISqrt( const I : LongWord) : Longword
5451:  Function ILog2( const I : LongWord) : LongWord
5452:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5453:  Function ESBArCosh( X : Extended) : Extended
5454:  Function ESBArSinh( X : Extended) : Extended
5455:  Function ESBArTanh( X : Extended) : Extended
5456:  Function ESBCosh( X : Extended) : Extended
5457:  Function ESBSinh( X : Extended) : Extended
5458:  Function ESBTanh( X : Extended) : Extended
5459:  Function InverseGamma( const X : Extended) : Extended
5460:  Function esbGamma( const X : Extended) : Extended
5461:  Function esbLnGamma( const X : Extended) : Extended
5462:  Function esbBeta( const X, Y : Extended) : Extended
5463:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5464: end;
5465:
5466: ******************************Integer Huge Cardinal Utils**************************
5467: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5468: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5469: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5470: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5471: Function BitCount_8( Value : byte) : integer
5472: Function BitCount_16( Value : uint16) : integer
5473: Function BitCount_32( Value : uint32) : integer
5474: Function BitCount_64( Value : uint64) : integer
5475: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5476:  Procedure ( CountPrimalityTests : integer)
5477:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5478:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5479:  Function isCoPrime( a, b : THugeCardinal) : boolean
5480:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5481:  Function hasSmallFactor( p : THugeCardinal) : boolean
5482:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer) : boolean
5483:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5484:  Const('StandardExponent','LongInt'( 65537);
5485: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5486:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5487:
5488: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5489: begin
```

```
5490:   AddTypeS('TXRTLInteger', 'array of Integer
5491:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5492:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5493:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5494:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5495:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5496:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5497:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5498:   'BitsPerByte','LongInt'( 8);
5499:   BitsPerDigit','LongInt'( 32);
5500:   SignBitMask','LongWord( $80000000);
5501:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5502:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5503:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5504:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5505:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5506:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5507:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5508:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5509:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5510:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5511:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5512:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5513:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5514:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5515:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5516:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5517:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5518:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5519:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5520:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5521:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5522:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5523:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5524:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5525:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5526:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5527:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5528:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5529:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5530:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5531:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5532:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5533:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5534:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5535:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5536:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5537:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5538:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger)
5539:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5540:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5541:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5542:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5543:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5544:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5545:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5546:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5547:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5548:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5549:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5550:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5551:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5552:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5553:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5554:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5555:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5556:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5557:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5558:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5559:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5560:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5561:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5562:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5563:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5564:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5565:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5566:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5567:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5568:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5569:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5570:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5571:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5572:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5573:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5574:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5575: end;
5576:
```

```
5577: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5578: begin
5579:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5580:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5581:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5582:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5583:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5584:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5585:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5586:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5587:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5588:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5589:  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)
5590: end;
5591:
5592:
5593: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5594: begin
5595:  Function StrDec( S : String) : String
5596:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5597:  Function ReadNumFromEdit( Edit : TEdit) : Float
5598:  Procedure WriteNumToFile( var F : Text; X : Float)
5599: end;
5600:
5601: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5602: begin
5603:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5604:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5605:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5606:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5607:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5608:  Procedure TeX_SetGraphTitle( Title : String)
5609:  Procedure TeX_SetOxTitle( Title : String)
5610:  Procedure TeX_SetOyTitle( Title : String)
5611:  Procedure TeX_PlotOxAxis
5612:  Procedure TeX_PlotOyAxis
5613:  Procedure TeX_PlotGrid( Grid : TGrid)
5614:  Procedure TeX_WriteGraphTitle
5615:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5616:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5617:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5618:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5619:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5620:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5621:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5622:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5623:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5624:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5625:  Function Xcm( X : Float) : Float
5626:  Function Ycm( Y : Float) : Float
5627: end;
5628:
5629: *---------------------------------------------------------------------------*)
5630: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5631: begin
5632:   TConstArray', 'array of TVarRec
5633:  Function CopyVarRec( const Item : TVarRec) : TVarRec
5634:  Function CreateConstArray( const Elements : array of const) : TConstArray
5635:  Procedure FinalizeVarRec( var Item : TVarRec)
5636:  Procedure FinalizeConstArray( var Arr : TConstArray)
5637: end;
5638:
5639: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5640: begin
5641:  Function HexBS( B : Byte) : ShortString
5642:  Function HexWS( W : Word) : ShortString
5643:  Function HexLS( L : LongInt) : ShortString
5644:  Function HexPtrS( P : Pointer) : ShortString
5645:  Function BinaryBS( B : Byte) : ShortString
5646:  Function BinaryWS( W : Word) : ShortString
5647:  Function BinaryLS( L : LongInt) : ShortString
5648:  Function OctalBS( B : Byte) : ShortString
5649:  Function OctalWS( W : Word) : ShortString
5650:  Function OctalLS( L : LongInt) : ShortString
5651:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5652:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5653:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5654:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5655:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5656:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5657:  Function Long2StrS( L : LongInt) : ShortString
5658:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
```

```
5659:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5660:  Function ValPrepS( const S : ShortString) : ShortString
5661:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5662:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5663:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5664:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5665:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5666:  Function TrimLeadS( const S : ShortString) : ShortString
5667:  Function TrimTrailS( const S : ShortString) : ShortString
5668:  Function TrimS( const S : ShortString) : ShortString
5669:  Function TrimSpacesS( const S : ShortString) : ShortString
5670:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5671:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5672:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5673:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5674:  Function ScrambleS( const S, Key : ShortString) : ShortString
5675:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5676:  Function FilterS( const S, Filters : ShortString) : ShortString
5677:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5678:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5679:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5680:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5681:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5682:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5683:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5684:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5685:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
5686:  Function CompStringS( const S1, S2 : ShortString) : Integer
5687:  Function CompUCStringS( const S1, S2 : ShortString) : Integer
5688:  Function SoundexS( const S : ShortString) : ShortString
5689:  Function MakeLetterSetS( const S : ShortString) : Longint
5690:  Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5691:  Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5692:  Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
      Pos:Cardinal):Bool;
5693:  Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5694:  Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5695:  Function JustFilenameS( const PathName : ShortString) : ShortString
5696:  Function JustNameS( const PathName : ShortString) : ShortString
5697:  Function JustExtensionS( const Name : ShortString) : ShortString
5698:  Function JustPathnameS( const PathName : ShortString) : ShortString
5699:  Function AddBackSlashS( const DirName : ShortString) : ShortString
5700:  Function CleanPathNameS( const PathName : ShortString) : ShortString
5701:  Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5702:  Function CommaizeS( L : LongInt) : ShortString
5703:  Function CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5704:  Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
      DecPt:Char):ShortString;
5705:  Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5706:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5707:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5708:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5709:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5710:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5711:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5712:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5713:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5714:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5715:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5716:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5717:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5718:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5719:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
      SubString:ShortString):Bool;
5720:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
      SubStr:ShortString):Bool;
5721:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5722:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5723:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5724:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5725:  Function ExtractTokensS(const S,
      Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5726:  Function IsChAlphaS( C : Char) : Boolean
5727:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5728:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5729:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5730:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5731:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5732:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5733:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5734:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5735:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5736:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5737:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
      Cardinal):ShortString;
```

```
5738:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
       Replacements:Cardinal):ShortString;
5739:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5740:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
       Replacements:Cardinal):ShortString
5741:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
       Replacements:Cardinal):ShortString
5742:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5743:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5744:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5745:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5746: end;
5747:
5748:
5749: *******unit uPSI_StUtils; from Systools4*****************************************************
5750: Function SignL( L : LongInt) : Integer
5751: Function SignF( F : Extended) : Integer
5752: Function MinWord( A, B : Word) : Word
5753: Function MidWord( W1, W2, W3 : Word) : Word
5754: Function MaxWord( A, B : Word) : Word
5755: Function MinLong( A, B : LongInt) : LongInt
5756: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5757: Function MaxLong( A, B : LongInt) : LongInt
5758: Function MinFloat( F1, F2 : Extended) : Extended
5759: Function MidFloat( F1, F2, F3 : Extended) : Extended
5760: Function MaxFloat( F1, F2 : Extended) : Extended
5761: Function MakeInteger16( H, L : Byte) : SmallInt
5762: Function MakeWordS( H, L : Byte) : Word
5763: Function SwapNibble( B : Byte) : Byte
5764: Function SwapWord( L : LongInt) : LongInt
5765: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5766: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5767: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5768: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5769: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5770: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5771: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5772: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5773: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5774: Procedure ExchangeBytes( var I, J : Byte)
5775: Procedure ExchangeWords( var I, J : Word)
5776: Procedure ExchangeLongInts( var I, J : LongInt)
5777: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5778: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5779: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5780: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5781: //******************uPSI_StFIN;*************************************************************
5782: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5783: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
       Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5784: Function BondDuration( Settlement,Maturity:TStDate;Rate,
       Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5785: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
       Extended
5786: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
       Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5787: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
       Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5788: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5789: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5790: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5791: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5792: Function DollarToDecimalText( DecDollar : Extended) : string
5793: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5794: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5795: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5796: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
       PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5797: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5798: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5799: Function InterestRateS(NPeriods:Int;Pmt,PV,
       FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5800: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5801: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5802: Function IsCardValid( const S : string) : Boolean
5803: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
       Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5804: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5805: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5806: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5807: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5808: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5809: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5810: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5811: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
       : TStPaymentTime): Extended
5812: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5813: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
       Timing: TStPaymentTime): Extended
```

```
5814: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5815: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5816: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5817: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5818: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5819: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5820: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5821: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5822: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5823:
5824: //*******************************************unit uPSI_StAstroP;
5825: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5826: //*****unit unit uPSI_StStat; Statistic Package of SysTools***************************************
5827: Function AveDev( const Data : array of Double) : Double
5828: Function AveDev16( const Data, NData : Integer) : Double
5829: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5830: Function Correlation( const Data1, Data2 : array of Double) : Double
5831: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5832: Function Covariance( const Data1, Data2 : array of Double) : Double
5833: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5834: Function DevSq( const Data : array of Double) : Double
5835: Function DevSq16( const Data, NData : Integer) : Double
5836: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5837:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5838: Function GeometricMeanS( const Data : array of Double) : Double
5839: Function GeometricMean16( const Data, NData : Integer) : Double
5840: Function HarmonicMeanS( const Data : array of Double) : Double
5841: Function HarmonicMean16( const Data, NData : Integer) : Double
5842: Function Largest( const Data : array of Double; K : Integer) : Double
5843: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5844: Function MedianS( const Data : array of Double) : Double
5845: Function Median16( const Data, NData : Integer) : Double
5846: Function Mode( const Data : array of Double) : Double
5847: Function Mode16( const Data, NData : Integer) : Double
5848: Function Percentile( const Data : array of Double; K : Double) : Double
5849: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5850: Function PercentRank( const Data : array of Double; X : Double) : Double
5851: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5852: Function Permutations( Number, NumberChosen : Integer) : Extended
5853: Function Combinations( Number, NumberChosen : Integer) : Extended
5854: Function FactorialS( N : Integer) : Extended
5855: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5856: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5857: Function Smallest( const Data : array of Double; K : Integer) : Double
5858: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5859: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5860: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5861:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5862:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5863: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5864: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5865: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5866: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5867: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5868: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5869: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5870: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5871: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5872: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5873: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5874: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5875: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5876: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5877: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5878: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5879: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5880: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5881: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5882: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5883: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5884: Function NormSDist( Z : Single) : Single
5885: Function NormSInv( Probability : Single) : Single
5886: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5887: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5888: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5889: Function Erfc( X : Single) : Single
5890: Function GammaLn( X : Single) : Single
5891: Function LargestSort( const Data : array of Double; K : Integer) : Double
5892: Function SmallestSort( const Data : array of double; K : Integer) : Double
5893:
5894: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5895:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5896:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5897:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5898:  Function DefaultMergeName( MergeNum : Integer) : string
```

```
5899:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5900:
5901: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5902: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5903:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5904:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5905:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5906:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5907:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5908:  Function LunarPhase( UT : TStDateTimeRec) : Double
5909:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5910:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5911:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5912:  Function FullMoon( D : TStDate) : TStLunarRecord
5913:  Function LastQuarter( D : TStDate) : TStLunarRecord
5914:  Function NewMoon( D : TStDate) : TStLunarRecord
5915:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5916:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5917:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5918:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5919:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5920:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5921:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5922:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5923:  Function SiderealTime( UT : TStDateTimeRec) : Double
5924:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5925:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5926:  Function SEaster( Y, Epoch : Integer) : TStDate
5927:  Function DateTimeToAJD( D : TDateTime) : Double
5928:  Function HoursMin( RA : Double) : ShortString
5929:  Function DegsMin( DC : Double) : ShortString
5930:  Function AJDToDateTime( D : Double) : TDateTime
5931:
5932: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5933:  Function CurrentDate : TStDate
5934:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5935:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5936:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5937:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5938:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5939:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5940:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5941:  Function WeekOfYear( Julian : TStDate) : Byte
5942:  Function AstJulianDate( Julian : TStDate) : Double
5943:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5944:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5945:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5946:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5947:  Function StIsLeapYear( Year : Integer) : Boolean
5948:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5949:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5950:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5951:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5952:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5953:  Function CurrentTime : TStTime
5954:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5955:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5956:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5957:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5958:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5959:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5960:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5961:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5962:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5963:  Function StDateToDateTime( D : TStDate) : TDateTime
5964:  Function StTimeToDateTime( T : TStTime) : TDateTime
5965:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5966:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5967:
5968: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5969: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5970:  Function MonthToString( const Month : Integer) : string
5971:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5972:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5973:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5974:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5975:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5976:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5977:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5978:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5979:  Function TimeStringToStTime( const Picture, S : string) : TStTime
5980:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5981:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5982:  Function DateStringIsBlank( const Picture, S : string) : Boolean
5983:  Function InternationalDate( ForceCentury : Boolean) : string
5984:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5985:  Function InternationalTime( ShowSeconds : Boolean) : string
5986:  Procedure ResetInternationalInfo
5987:
```

```
5988: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5989:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5990:  Function AnsiUpperCaseShort32( const S : string) : string
5991:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
5992:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
5993:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
5994:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5995:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5996:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5997:  Function Upcase( C : AnsiChar) : AnsiChar
5998:  Function LoCase( C : AnsiChar) : AnsiChar
5999:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
6000:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
6001:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6002:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6003:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6004:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
6005:  Procedure RaiseContainerError( Code : longint)
6006:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
6007:  Function ProductOverflow( A, B : LongInt) : Boolean
6008:  Function StNewStr( S : string) : PShortString
6009:  Procedure StDisposeStr( PS : PShortString)
6010:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
6011:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
6012:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
6013:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6014:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6015:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
6016:
6017: procedure SIRegister_usvd(CL: TPSPascalCompiler);
6018: begin
6019:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
6020:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
6021:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
6022:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
6023:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6024: end;
6025:
6026: //**********unit unit ; StMath Package of SysTools*****************************************
6027: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6028: Function PowerS( Base, Exponent : Extended) : Extended
6029: Function StInvCos( X : Double) : Double
6030: Function StInvSin( Y : Double) : Double
6031: Function StInvTan2( X, Y : Double) : Double
6032: Function StTan( A : Double) : Double
6033: Procedure DumpException;  //unit StExpEng;
6034: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6035:
6036: //**********unit unit ; StCRC Package of SysTools*****************************************
6037: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6038: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6039: Function Adler32OfFile( FileName : AnsiString) : LongInt
6040: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6041: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6042: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6043: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6044: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6045: Function Crc32OfFile( FileName : AnsiString) : LongInt
6046: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6047: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6048: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6049: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6050: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6051: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6052:
6053: //**********unit unit ; StBCD Package of SysTools*****************************************
6054: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6055: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6056: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6057: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6058: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6059: Function NegBcd( const B : TbcdS) : TbcdS
6060: Function AbsBcd( const B : TbcdS) : TbcdS
6061: Function FracBcd( const B : TbcdS) : TbcdS
6062: Function IntBcd( const B : TbcdS) : TbcdS
6063: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6064: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6065: Function ValBcd( const S : string) : TbcdS
6066: Function LongBcd( L : LongInt) : TbcdS
6067: Function ExtBcd( E : Extended) : TbcdS
6068: Function ExpBcd( const B : TbcdS) : TbcdS
6069: Function LnBcd( const B : TbcdS) : TbcdS
6070: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6071: Function PowBcd( const B, E : TbcdS) : TbcdS
6072: Function SqrtBcd( const B : TbcdS) : TbcdS
6073: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6074: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6075: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6076: Function IsIntBcd( const B : TbcdS) : Boolean
```

```
6077: Function TruncBcd( const B : TbcdS) : LongInt
6078: Function BcdExt( const B : TbcdS) : Extended
6079: Function RoundBcd( const B : TbcdS) : LongInt
6080: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6081: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6082: Function FormatBcd( const Format : string; const B : TbcdS) : string
6083: Function StrGeneralBcd( const B : TbcdS) : string
6084: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6085: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6086:
6087: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6088: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6089: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6090: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6091: Function StDeEscape( const EscStr : AnsiString) : Char
6092: Function StDoEscape( Delim : Char) : AnsiString
6093: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6094: Function AnsiHashText( const S : string; Size : Integer) : Integer
6095: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6096: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6097: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6098:
6099: //**********unit unit ; StNetCon Package of SysTools*****************************************
6100:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6101:     Constructor Create( AOwner : TComponent)
6102:     Function Connect : DWord
6103:     Function Disconnect : DWord
6104:     RegisterProperty('Password', 'String', iptrw);
6105:     Property('UserName', 'String', iptrw);
6106:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6107:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6108:     Property('LocalDevice', 'String', iptrw);
6109:     Property('ServerName', 'String', iptrw);
6110:     Property('ShareName', 'String', iptrw);
6111:     Property('OnConnect', 'TNotifyEvent', iptrw);
6112:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6113:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6114:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6115:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6116:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6117:   end;
6118: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6119: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6120: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6121: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6122: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6123: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6124: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6125: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6126: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6127: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6128: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6129: Function SuspendThread( hThread : THandle) : DWORD
6130: Function ResumeThread( hThread : THandle) : DWORD
6131: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6132: Function GetCurrentThread : THandle
6133: Procedure ExitThread( dwExitCode : DWORD)
6134: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6135: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6136: Procedure EndThread(ExitCode: Integer);
6137: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6138: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6139: Procedure FreeProcInstance( Proc : FARPROC)
6140: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6141: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6142: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6143: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6144: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6145: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6146: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6147: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6148: Function CurrentParallelJobInfo : TParallelJobInfo
6149: Function ObtainParallelJobInfo : TParallelJobInfo
6150: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6151: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6152: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6153: Function
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL');
6154: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL');
6155: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD) : BOOL');
6156: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6157: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6158:
6159: ************************************************unit uPSI_JclMime;
6160:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6161:  Function MimeDecodeString( const S : AnsiString) : AnsiString
```

```
6162:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6163:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6164:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6165:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6166:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6167:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6168:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6169:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6170:
6171: ************************************************unit uPSI_JclPrint;
6172:  Procedure DirectPrint( const Printer, Data : string)
6173:  Procedure SetPrinterPixelsPerInch
6174:  Function GetPrinterResolution : TPoint
6175:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6176:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6177:
6178:
6179: //************************************unit uPSI_ShLwApi;***********************************
6180:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6181:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6182:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6183:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6184:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6185:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6186:  Function StrDup( lpSrch : PChar) : PChar
6187:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6188:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6189:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6190:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6191: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6192:  Function StrPBrk( psz, pszSet : PChar) : PChar
6193:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6194:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6195:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6196:  Function StrSpn( psz, pszSet : PChar) : Integer
6197:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6198:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6199:  Function StrToInt( lpSrch : PChar) : Integer
6200:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6201:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6202:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6203:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6204:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6205:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6206:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6207:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6208:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6209:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6210:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6211:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6212:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6213:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6214:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6215:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6216:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6217:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6218:  STIF_DEFAULT','LongWord( $00000000);
6219:  STIF_SUPPORT_HEX','LongWord( $00000001);
6220:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6221:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6222:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6223:  Function PathAddBackslash( pszPath : PChar) : PChar
6224:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6225:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6226:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6227:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6228:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6229:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6230:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6231:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6232:  Function PathFileExists( pszPath : PChar) : BOOL
6233:  Function PathFindExtension( pszPath : PChar) : PChar
6234:  Function PathFindFileName( pszPath : PChar) : PChar
6235:  Function PathFindNextComponent( pszPath : PChar) : PChar
6236:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6237:  Function PathGetArgs( pszPath : PChar) : PChar
6238:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6239:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6240:  Function PathGetCharType( ch : Char) : UINT
6241:  GCT_INVALID','LongWord( $0000);
6242:  GCT_LFNCHAR','LongWord( $0001);
6243:  GCT_SHORTCHAR','LongWord( $0002);
6244:  GCT_WILD','LongWord( $0004);
6245:  GCT_SEPARATOR','LongWord( $0008);
6246:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6247:  Function PathIsDirectory( pszPath : PChar) : BOOL
6248:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
```

```
6249:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6250:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6251:  Function PathIsRelative( pszPath : PChar) : BOOL
6252:  Function PathIsRoot( pszPath : PChar) : BOOL
6253:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6254:  Function PathIsUNC( pszPath : PChar) : BOOL
6255:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6256:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6257:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6258:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6259:  Function PathIsURL( pszPath : PChar) : BOOL
6260:  Function PathMakePretty( pszPath : PChar) : BOOL
6261:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6262:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6263:  Procedure PathQuoteSpaces( lpsz : PChar)
6264:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6265:  Procedure PathRemoveArgs( pszPath : PChar)
6266:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6267:  Procedure PathRemoveBlanks( pszPath : PChar)
6268:  Procedure PathRemoveExtension( pszPath : PChar)
6269:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6270:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6271:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6272:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6273:  Function PathSkipRoot( pszPath : PChar) : PChar
6274:  Procedure PathStripPath( pszPath : PChar)
6275:  Function PathStripToRoot( pszPath : PChar) : BOOL
6276:  Procedure PathUnquoteSpaces( lpsz : PChar)
6277:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6278:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6279:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6280:  Procedure PathUndecorate( pszPath : PChar)
6281:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6282:  URL_SCHEME_INVALID','LongInt'( - 1);
6283:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6284:  URL_SCHEME_FTP','LongInt'( 1);
6285:  URL_SCHEME_HTTP','LongInt'( 2);
6286:  URL_SCHEME_GOPHER','LongInt'( 3);
6287:  URL_SCHEME_MAILTO','LongInt'( 4);
6288:  URL_SCHEME_NEWS','LongInt'( 5);
6289:  URL_SCHEME_NNTP','LongInt'( 6);
6290:  URL_SCHEME_TELNET','LongInt'( 7);
6291:  URL_SCHEME_WAIS','LongInt'( 8);
6292:  URL_SCHEME_FILE','LongInt'( 9);
6293:  URL_SCHEME_MK','LongInt'( 10);
6294:  URL_SCHEME_HTTPS','LongInt'( 11);
6295:  URL_SCHEME_SHELL','LongInt'( 12);
6296:  URL_SCHEME_SNEWS','LongInt'( 13);
6297:  URL_SCHEME_LOCAL','LongInt'( 14);
6298:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6299:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6300:  URL_SCHEME_ABOUT','LongInt'( 17);
6301:  URL_SCHEME_RES','LongInt'( 18);
6302:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6303:  URL_SCHEME', 'Integer
6304:  URL_PART_NONE','LongInt'( 0);
6305:  URL_PART_SCHEME','LongInt'( 1);
6306:  URL_PART_HOSTNAME','LongInt'( 2);
6307:  URL_PART_USERNAME','LongInt'( 3);
6308:  URL_PART_PASSWORD','LongInt'( 4);
6309:  URL_PART_PORT','LongInt'( 5);
6310:  URL_PART_QUERY','LongInt'( 6);
6311:  URL_PART', 'DWORD
6312:  URLIS_URL','LongInt'( 0);
6313:  URLIS_OPAQUE','LongInt'( 1);
6314:  URLIS_NOHISTORY','LongInt'( 2);
6315:  URLIS_FILEURL','LongInt'( 3);
6316:  URLIS_APPLIABLE','LongInt'( 4);
6317:  URLIS_DIRECTORY','LongInt'( 5);
6318:  URLIS_HASQUERY','LongInt'( 6);
6319:  TUrlIs', 'DWORD
6320:  URL_UNESCAPE','LongWord( $10000000);
6321:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6322:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6323:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6324:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6325:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6326:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6327:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6328:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6329:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6330:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6331:  URL_INTERNAL_PATH','LongWord( $00800000);
6332:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6333:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6334:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6335:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6336:  URL_APPLY_DEFAULT','LongWord( $00000001);
6337:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
```

```
6338:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6339:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6340:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6341:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6342:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6343:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6344:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6345:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6346:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6347:  Function UrlGetLocation( psz1 : PChar) : PChar
6348:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6349:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6350:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6351:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6352:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6353:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6354:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6355:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6356:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6357:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6358:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6359:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6360:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6361:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6362:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
      pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6363:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6364:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6365:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6366:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6367:  SHREGDEL_DEFAULT','LongWord( $00000000);
6368:  SHREGDEL_HKCU','LongWord( $00000001);
6369:  SHREGDEL_HKLM','LongWord( $00000010);
6370:  SHREGDEL_BOTH','LongWord( $00000011);
6371:  SHREGENUM_DEFAULT','LongWord( $00000000);
6372:  SHREGENUM_HKCU','LongWord( $00000001);
6373:  SHREGENUM_HKLM','LongWord( $00000010);
6374:  SHREGENUM_BOTH','LongWord( $00000011);
6375:  SHREGSET_HKCU','LongWord( $00000001);
6376:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6377:  SHREGSET_HKLM','LongWord( $00000004);
6378:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6379:  TSHRegDelFlags', 'DWORD
6380:  TSHRegEnumFlags', 'DWORD
6381:  HUSKEY', 'THandle
6382:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6383:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6384:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6385:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6386:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6387:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6388:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6389:  ASSOCF_VERIFY','LongWord( $00000040);
6390:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6391:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6392:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6393:  ASSOCF', 'DWORD
6394:  ASSOCSTR_COMMAND','LongInt'( 1);
6395:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6396:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6397:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6398:  ASSOCSTR_NOOPEN','LongInt'( 5);
6399:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6400:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6401:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6402:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6403:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6404:  ASSOCSTR_INFOTIP','LongInt'( 11);
6405:  ASSOCSTR_MAX','LongInt'( 12);
6406:  ASSOCSTR', 'DWORD
6407:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6408:  ASSOCKEY_APP','LongInt'( 2);
6409:  ASSOCKEY_CLASS','LongInt'( 3);
6410:  ASSOCKEY_BASECLASS','LongInt'( 4);
6411:  ASSOCKEY_MAX','LongInt'( 5);
6412:  ASSOCKEY', 'DWORD
6413:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6414:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6415:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6416:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6417:  ASSOCDATA_MAX','LongInt'( 5);
6418:  ASSOCDATA', 'DWORD
6419:  ASSOCENUM_NONE','LongInt'( 0);
6420:  ASSOCENUM', 'DWORD
6421:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6422:  SHACF_DEFAULT $00000000);
6423:  SHACF_FILESYSTEM','LongWord( $00000001);
6424:  SHACF_URLHISTORY','LongWord( $00000002);
6425:  SHACF_URLMRU','LongWord( $00000004);
```

```
6426:  SHACF_USETAB','LongWord( $00000008);
6427:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6428:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6429:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6430:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6431:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6432:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6433:  Procedure SHSetThreadRef( punk : IUnknown)
6434:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6435:  CTF_INSIST','LongWord( $00000001);
6436:  CTF_THREAD_REF','LongWord( $00000002);
6437:  CTF_PROCESS_REF','LongWord( $00000004);
6438:  CTF_COINIT','LongWord( $00000008);
6439:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6440:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6441:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6442:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6443:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6444:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6445:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6446:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6447:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6448:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6449:  Function SetRectEmpty( var lprc : TRect) : BOOL
6450:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6451:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6452:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6453:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6454:
6455:  Function InitializeFlatSB( hWnd : HWND) : Bool
6456:  Procedure UninitializeFlatSB( hWnd : HWND)
6457:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6458:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6459:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6460:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6461:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6462:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6463:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6464:
6465:
6466: //  ***************************************** 204 unit uPSI_ShellAPI;
6467: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6468: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6469: Procedure DragFinish( Drop : HDROP)
6470: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6471: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6472: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6473: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6474: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6475: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6476: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6477: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6478: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6479: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6480: Procedure SHFreeNameMappings( hNameMappings : THandle)
6481:
6482:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6483:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6484:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6485:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6486:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6487:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6488:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6489:  Function SimpleXMLEncode( const S : string) : string
6490:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6491:  Function XMLEncode( const S : string) : string
6492:  Function XMLDecode( const S : string) : string
6493:  Function EntityEncode( const S : string) : string
6494:  Function EntityDecode( const S : string) : string
6495:
6496: procedure RIRegister_CPort_Routines(S: TPSExec);
6497:  Procedure EnumComPorts( Ports : TStrings)
6498:  Procedure ListComPorts( Ports : TStrings)
6499:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6500:  Function GetComPorts: TStringlist;
6501:  Function StrToBaudRate( Str : string) : TBaudRate
6502:  Function StrToStopBits( Str : string) : TStopBits
6503:  Function StrToDataBits( Str : string) : TDataBits
6504:  Function StrToParity( Str : string) : TParityBits
6505:  Function StrToFlowControl( Str : string) : TFlowControl
6506:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6507:  Function StopBitsToStr( StopBits : TStopBits) : string
6508:  Function DataBitsToStr( DataBits : TDataBits) : string
6509:  Function ParityToStr( Parity : TParityBits) : string
6510:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6511:  Function ComErrorsToStr( Errors : TComErrors) : String
6512:
6513:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6514:  Function DispatchMessage( const lpMsg : TMsg) : Longint
```

```
6515:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6516:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6517:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6518:  Function GetMessagePos : DWORD
6519:  Function GetMessageTime : Longint
6520:  Function GetMessageExtraInfo : Longint
6521:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6522:  Procedure JAddToRecentDocs( const Filename : string)
6523:  Procedure ClearRecentDocs
6524:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6525:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6526:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6527:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6528:  Function RecycleFile( FileToRecycle : string) : Boolean
6529:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6530:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6531:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6532:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6533:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6534:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6535:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6536:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6537:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6538:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6539:
6540: ****************************************** unit uPSI_JclPeImage;
6541:
6542:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6543: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6544:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6545:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
6546:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6547:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6548:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6549:  Function PeDoesExportFunction(const FileName:TFileName;const
      FuncName:string;Options:TJclSmartCompOptions):Bool;
6550:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6551:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
      FunctionName:string;Options:TJclSmartCompOptions):Bool
6552:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean
6553:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
6554:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean
6555:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
      LibraryName:string; IncludeLibNames : Boolean): Boolean
6556:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6557:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6558:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6559:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
      NamesList:TStrings):Bool
6560:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6561:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
      Descript:Bool):Bool;
6562:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6563:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6564:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6565:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6566:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6567:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6568:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
      PImageSectionHeader
6569:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6570:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6571:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
      ___Pointer;
6572:    SIRegister_TJclPeSectionStream(CL);
6573:    SIRegister_TJclPeMapImgHookItem(CL);
6574:    SIRegister_TJclPeMapImgHooks(CL);
6575:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
      NtHeaders:TImageNtHeaders):Boolean
6576:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6577:    Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6578:    TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6579:    TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6580:    TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6581:    TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
```

```
6582:   TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6583:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
      TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6584:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
      Descript:TJclBorUmDescription):TJclBorUmResult;
6585:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6586:  Function PeBorUnmangleName3( const Name : string) : string;
6587:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6588:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6589:
6590:
6591: //***************** SysTools uPSI_StSystem; ************************************
6592:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6593:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6594:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6595:  //Procedure EnumerateDirectories(const
      StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6596:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
      IncludeItem:TIncludeItemFunc);
6597:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6598:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6599:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6600:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6601:  Function FlushOsBuffers( Handle : Integer) : Boolean
6602:  Function GetCurrentUser : AnsiString
6603:  Function GetDiskClass( Drive : Char) : DiskClass
6604:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
      SectorsPerCluster:Cardinal):Bool;
6605:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
      DiskSize:Double):Bool;
6606:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
      DiskSize:Comp):Boolean;
6607:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6608:  Function getDiskSpace2(const path: String; index: integer): int64;
6609:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6610:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6611:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6612:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6613:  Function GetLongPath( const APath : AnsiString) : AnsiString
6614:  Function GetMachineName : AnsiString
6615:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6616:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6617:  Function GetShortPath( const APath : AnsiString) : AnsiString
6618:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6619:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6620:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6621:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6622:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6623:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6624:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6625:  Function IsDriveReady( Drive : Char) : Boolean
6626:  Function IsFile( const FileName : AnsiString) : Boolean
6627:  Function IsFileArchive( const S : AnsiString) : Integer
6628:  Function IsFileHidden( const S : AnsiString) : Integer
6629:  Function IsFileReadOnly( const S : AnsiString) : Integer
6630:  Function IsFileSystem( const S : AnsiString) : Integer
6631:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6632:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6633:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6634:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6635:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6636:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6637:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6638:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6639:  Function ValidDrive( Drive : Char) : Boolean
6640:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6641:
6642: //****************************unit uPSI_JclLANMan;*************************************
6643: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6644:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6645:  Function DeleteAccount( const Servername, Username : string) : Boolean
6646:  Function DeleteLocalAccount( Username : string) : Boolean
6647:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6648:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6649:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6650:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6651:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6652:  Function LocalGroupExists( const Group : string) : Boolean
6653:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6654:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6655:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6656:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6657:  Function IsLocalAccount( const AccountName : string) : Boolean
6658:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6659:  Function GetRandomString( NumChar : cardinal) : string
6660:
6661: //****************************unit uPSI_cUtils;*****************************************
```

```
6662: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6663: Function cIsWinNT : boolean
6664:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6665:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6666:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6667:  Function cGetShortName( FileName : string) : string
6668:  Procedure cShowError( Msg : String)
6669:  Function cCommaStrToStr( s : string; formatstr : string) : string
6670:  Function cIncludeQuoteIfSpaces( s : string) : string
6671:  Function cIncludeQuoteIfNeeded( s : string) : string
6672:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6673:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6674:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6675:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6676:  Function cCodeInstoStr( s : string) : string
6677:  Function cStrtoCodeIns( s : string) : string
6678:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6679:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6680:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6681:  Function cPointtoStr( const pt : TPoint) : string
6682:  Function cListtoStr( const List : TStrings) : string
6683:  Function ListtoStr( const List : TStrings) : string
6684:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6685:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6686:  Function cGetFileTyp( const FileName : string) : TUnitType
6687:  Function cGetExTyp( const FileName : string) : TExUnitType
6688:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6689:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6690:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6691:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6692:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6693:  Function cGenMakePath( FileName : String) : String;
6694:  Function cGenMakePath2( FileName : String) : String
6695:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6696:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6697:  Function cCalcMod( Count : Integer) : Integer
6698:  Function cGetVersionString( FileName : string) : string
6699:  Function cCheckChangeDir( var Dir : string) : boolean
6700:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6701:  Function cIsNumeric( s : string) : boolean
6702:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6703:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6704:  Function GetFileTyp( const FileName : string) : TUnitType
6705:  Function Atoi(const aStr: string): integer
6706:  Function Itoa(const aint: integer): string
6707:  Function Atof(const aStr: string): double');
6708:  Function Atol(const aStr: string): longint');
6709:
6710:
6711: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6712: begin
6713:   FindClass('TOBJECT'),'EHTTP
6714:   FindClass('TOBJECT'),'EHTTPParser
6715:   //AnsiCharSet', 'set of AnsiChar
6716:   AnsiStringArray', 'array of AnsiString
6717:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6718:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6719:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6720:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6721:    +'CustomMinVersion : Integer; end
6722:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6723:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6724:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6725:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6726:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6727:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6728:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6729:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6730:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6731:    +'nection, hntOrigin, hntKeepAlive )
6732:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6733:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6734:    +' AnsiString; end
6735:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6736:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6737:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6738:   //PHTTPContentLength', '^THTTPContentLength // will not work
6739:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6740:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6741:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6742:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6743:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6744:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6745:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6746:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6747:    +' CustomStr : AnsiString; end
6748:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
```

```
6749:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6750:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6751:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6752:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6753:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6754:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6755:    +'m; Custom : AnsiString; end
6756:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6757:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6758:    +' Custom : AnsiString; end
6759:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6760:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6761:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6762:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6763:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6764:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6765:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6766:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6767:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6768:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6769:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6770:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6771:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6772:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6773:    +'FieldEnum; List : array of THTTPContentEncoding; end
6774:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6775:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6776:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6777:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6778:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6779:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6780:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6781:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6782:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6783:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6784:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6785:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6786:    +'CustomFieldArray; Custom : AnsiString; end
6787:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6788:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6789:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6790:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6791:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6792:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6793:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6794:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6795:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6796:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6797:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6798:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6799:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6800:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6801:   THTTPFixedHeaders','array[0..42] of AnsiString
6802:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6803:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6804:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6805:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6806:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6807:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6808:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6809:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6810:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6811:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6812:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6813:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6814:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6815:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6816:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6817:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6818:    +' THTTPDateField; Age : THTTPAgeField; end
6819:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6820:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6821:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6822:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6823:  Procedure InitHTTPRequest( var A : THTTPRequest)
6824:  Procedure InitHTTPResponse( var A : THTTPResponse)
6825:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6826:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6827:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6828:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6829:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6830:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6831:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6832:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6833:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6834:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6835:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6836:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6837:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
```

```
6838:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6839:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6840:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6841:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6842:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6843:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6844:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6845:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6846:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6847:   THTTPStringOption', '( hsoNone )
6848:   THTTPStringOptions', 'set of THTTPStringOption
6849:   FindClass('TOBJECT'),'TAnsiStringBuilder
6850:
6851:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6852:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6853:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6854:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6855:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6856:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6857:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6858:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6859:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6860:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6861:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6862:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6863:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6864:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6865:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6866:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
6867:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6868:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6869:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions)
6870:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6871:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6872:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6873:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6874:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6875:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6876:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6877:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6878:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6879:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
      THTTPStrOptions);
6880:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6881:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
      P:THTTPStringOptions);
6882:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6883:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6884:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6885:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6886:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6887:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6888:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
      Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6889:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6890:   SIRegister_THTTPParser(CL);
6891:   FindClass('TOBJECT'),'THTTPContentDecoder
6892:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6893:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6894:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6895:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
```

```
6896:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6897:   SIRegister_THTTPContentDecoder(CL);
6898:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6899:   FindClass('TOBJECT'),'THTTPContentReader
6900:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6901:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
      LogLevel:Int;
6902:   SIRegister_THTTPContentReader(CL);
6903:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6904:   FindClass('TOBJECT'),'THTTPContentWriter
6905:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6906:   SIRegister_THTTPContentWriter(CL);
6907:  Procedure SelfTestcHTTPUtils
6908: end;
6909:
6910: (*----------------------------------------------------------------------------*)
6911: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6912: begin
6913:  'TLSLibraryVersion','String '1.00
6914:  'TLSError_None','LongInt'( 0);
6915:  'TLSError_InvalidBuffer','LongInt'( 1);
6916:  'TLSError_InvalidParameter','LongInt'( 2);
6917:  'TLSError_InvalidCertificate','LongInt'( 3);
6918:  'TLSError_InvalidState','LongInt'( 4);
6919:  'TLSError_DecodeError','LongInt'( 5);
6920:  'TLSError_BadProtocol','LongInt'( 6);
6921:  Function TLSErrorMessage( const TLSError : Integer) : String
6922:   SIRegister_ETLSError(CL);
6923:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6924:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6925:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6926:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6927:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6928:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6929:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6930:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6931:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6932:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6933:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6934:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6935:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6936:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6937:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6938:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6939:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6940:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6941:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6942:   PTLSRandom', '^TTLSRandom // will not work
6943:  Procedure InitTLSRandom( var Random : TTLSRandom)
6944:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6945:  'TLSSessionIDMaxLen','LongInt'( 32);
6946:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6947:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6948:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6949:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6950:    +'; Signature : TTLSSignatureAlgorithm; end
6951:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6952:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6953:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6954:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6955:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6956:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6957:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6958:    +'nteger; Supported : Boolean; end
6959:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6960:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
6961:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6962:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6963:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6964:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6965:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6966:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6967:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6968:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6969:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
6970:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
6971:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6972:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6973:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
6974:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6975:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6976:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6977:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
```

```
6978:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6979:    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6980:    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6981:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
       IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6982:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
       ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6983:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
6984:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
6985:  Procedure SelfTestcTLSUtils
6986: end;
6987:
6988: (*----------------------------------------------------------------------------*)
6989: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6990: begin
6991:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
       integer; disks : integer; mx : integer; my : integer; end
6992:  // pBoard', '^tBoard // will not work
6993:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6994:  Function rCheckMove( color : byte; cx, cy : integer) : integer
6995:  //Function rDoStep( data : pBoard) : word
6996:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6997: end;
6998:
6999: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7000: begin
7001: Function InEditMode( ADataset : TDataset) : Boolean
7002: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
7003: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7004: Function GetFieldText( AField : TField) : String
7005: end;
7006:
7007: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
7008: begin
7009:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7010:   TMyPrintRange', '( prAll, prSelected )
7011:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7012:    +'ded, ssDateTime, ssTime, ssCustom )
7013:   TSortDirection', '( sdAscending, sdDescending )
7014:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7015:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7016:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7017:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7018:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7019:   SIRegister_TSortOptions(CL);
7020:   SIRegister_TPrintOptions(CL);
7021:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7022:   SIRegister_TSortedList(CL);
7023:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7024:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7025:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7026:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7027:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7028:   SIRegister_TFontSetting(CL);
7029:   SIRegister_TFontList(CL);
7030:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
7031:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7032:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7033:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7034:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
7035:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7036:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7037:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7038:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
7039:    +'r; var SortStyle : TSortStyle)
7040:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
7041:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7042:   SIRegister_TSortGrid(CL);
7043:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
7044:  Function NormalCompare( const Str1, Str2 : String) : Integer
7045:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
7046:  Function NumericCompare( const Str1, Str2 : String) : Integer
7047:  Function TimeCompare( const Str1, Str2 : String) : Integer
7048:  //Function Compare( Item1, Item2 : Pointer) : Integer
7049: end;
7050:
7051: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
7052:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
7053:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7054:  Procedure IBDataBaseError
7055:  Function StatusVector : PISC_STATUS
7056:  Function StatusVectorArray : PStatusVector
7057:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7058:  Function StatusVectorAsText : string
7059:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7060:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7061:
7062:
7063: //****************************unit uPSI_BoldUtils;****************************************
```

```
7064:  Function CharCount( c : char; const s : string) : integer
7065:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7066:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7067:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7068:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7069:  Function BoldTrim( const S : string) : string
7070:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7071:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7072:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7073:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7074:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7075:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7076:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7077:  Function CapitalisedToSpaced( Capitalised : String) : String
7078:  Function SpacedToCapitalised( Spaced : String) : String
7079:  Function BooleanToString( BoolValue : Boolean) : String
7080:  Function StringToBoolean( StrValue : String) : Boolean
7081:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7082:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7083:  Function StringListToVarArray( List : TStringList) : variant
7084:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7085:  Function GetComputerNameStr : string
7086:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7087:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7088:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7089:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7090:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7091:  Procedure EnsureTrailing( var Str : String; ch : char)
7092:  Function BoldDirectoryExists( const Name : string) : Boolean
7093:  Function BoldForceDirectories( Dir : string) : Boolean
7094:  Function BoldRootRegistryKey : string
7095:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7096:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7097:  Function LogicalAnd( A, B : Integer) : Boolean
7098:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7099:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7100:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7101:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7102:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7103:  Function IsFirstInstance : Boolean
7104:  Procedure ActivateFirst( AString : PChar)
7105:  Procedure ActivateFirstCommandLine
7106:  function MakeAckPkt(const BlockNumber: Word): string;
7107:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7108:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7109:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7110:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7111:  function IdStrToWord(const Value: String): Word;
7112:  function IdWordToStr(const Value: Word): WordStr;
7113:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7114:  Function CPUFeatures : TCPUFeatures
7115:
7116: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7117: begin
7118:   AddTypeS('TXRTLBitIndex', 'Integer
7119:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7120:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7121:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7122:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7123:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7124:  Function XRTLSwapHiLo16( X : Word) : Word
7125:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7126:  Function XRTLSwapHiLo64( X : Int64) : Int64
7127:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7128:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7129:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7130:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7131:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7132:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7133:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7134:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7135:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7136:  Function XRTLPopulation( A : Cardinal) : Cardinal
7137: end;
7138:
7139: Function XRTLURLDecode( const ASrc : WideString) : WideString
7140: Function XRTLURLEncode( const ASrc : WideString) : WideString
7141: Function XRTLURINormalize( const AURI : WideString) : WideString
7142: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7143: Function XRTLExtractLongPathName(APath: string): string;
7144:
7145: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7146: begin
7147:   AddTypeS('Int8', 'ShortInt
7148:   AddTypeS('Int16', 'SmallInt
7149:   AddTypeS('Int32', 'LongInt
7150:   AddTypeS('UInt8', 'Byte
```

```
7151:   AddTypeS('UInt16', 'Word
7152:   AddTypeS('UInt32', 'LongWord
7153:   AddTypeS('UInt64', 'Int64
7154:   AddTypeS('Word8', 'UInt8
7155:   AddTypeS('Word16', 'UInt16
7156:   AddTypeS('Word32', 'UInt32
7157:   AddTypeS('Word64', 'UInt64
7158:   AddTypeS('LargeInt', 'Int64
7159:   AddTypeS('NativeInt', 'Integer
7160:   AddTypeS('NativeUInt', 'Cardinal
7161:   Const('BitsPerByte','LongInt'( 8);
7162:   Const('BitsPerWord','LongInt'( 16);
7163:   Const('BitsPerLongWord','LongInt'( 32);
7164:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7165:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7166:  Function MinI( const A, B : Integer) : Integer
7167:  Function MaxI( const A, B : Integer) : Integer
7168:  Function MinC( const A, B : Cardinal) : Cardinal
7169:  Function MaxC( const A, B : Cardinal) : Cardinal
7170:  Function SumClipI( const A, I : Integer) : Integer
7171:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7172:  Function InByteRange( const A : Int64) : Boolean
7173:  Function InWordRange( const A : Int64) : Boolean
7174:  Function InLongWordRange( const A : Int64) : Boolean
7175:  Function InShortIntRange( const A : Int64) : Boolean
7176:  Function InSmallIntRange( const A : Int64) : Boolean
7177:  Function InLongIntRange( const A : Int64) : Boolean
7178:   AddTypeS('Bool8', 'ByteBool
7179:   AddTypeS('Bool16', 'WordBool
7180:   AddTypeS('Bool32', 'LongBool
7181:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7182:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7183:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7184:  Const('MinSingle','Single').setExtended( 1.5E-45);
7185:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7186:  Const('MinDouble','Double').setExtended( 5.0E-324);
7187:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7188:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7189:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7190:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7191:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7192:  Function MinF( const A, B : Float) : Float
7193:  Function MaxF( const A, B : Float) : Float
7194:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7195:  Function InSingleRange( const A : Float) : Boolean
7196:  Function InDoubleRange( const A : Float) : Boolean
7197:  Function InCurrencyRange( const A : Float) : Boolean;
7198:  Function InCurrencyRange1( const A : Int64) : Boolean;
7199:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7200:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7201:  Function FloatIsInfinity( const A : Extended) : Boolean
7202:  Function FloatIsNaN( const A : Extended) : Boolean
7203:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7204:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7205:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7206:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7207:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7208:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7209:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7210:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7211:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7212:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7213:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7214:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7215:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7216:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7217:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7218:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7219:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7220:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7221:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7222:  Function SetBitScanForward( const Value : LongWord) : Integer;
7223:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7224:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7225:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7226:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7227:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7228:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7229:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7230:  Function cReverseBits( const Value : LongWord) : LongWord;
7231:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7232:  Function cSwapEndian( const Value : LongWord) : LongWord
7233:  Function cTwosComplement( const Value : LongWord) : LongWord
7234:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7235:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7236:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7237:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7238:  Function cBitCount( const Value : LongWord) : LongWord
7239:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
```

```
7240:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7241:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7242:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7243:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7244:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7245:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7246:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7247:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7248: //  AddTypeS('CharSet', 'set of AnsiChar
7249:   AddTypeS('CharSet', 'set of Char       //!!!
7250:   AddTypeS('AnsiCharSet', 'TCharSet
7251:   AddTypeS('ByteSet', 'set of Byte
7252:   AddTypeS('AnsiChar', 'Char
7253:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7254:  Function AsByteSet( const C : array of Byte) : ByteSet
7255:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7256:  Procedure ClearCharSet( var C : CharSet)
7257:  Procedure FillCharSet( var C : CharSet)
7258:  procedure FillCharSearchRec;  // with 0
7259:  Procedure ComplementCharSet( var C : CharSet)
7260:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7261:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7262:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7263:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7264:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7265:  Function IsSubSet( const A, B : CharSet) : Boolean
7266:  Function IsEqual( const A, B : CharSet) : Boolean
7267:  Function IsEmpty( const C : CharSet) : Boolean
7268:  Function IsComplete( const C : CharSet) : Boolean
7269:  Function cCharCount( const C : CharSet) : Integer
7270:  Procedure ConvertCaseInsensitive( var C : CharSet)
7271:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7272:  Function IntRangeLength( const Low, High : Integer) : Int64
7273:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7274:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7275:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7276:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7277:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7278:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7279:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7280:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7281:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7282:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7283:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7284:   AddTypeS('UnicodeChar', 'WideChar
7285:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7286:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7287:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7288:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7289:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7290:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7291:  Function cSgn( const A : LongInt) : Integer;
7292:  Function cSgn1( const A : Int64) : Integer;
7293:  Function cSgn2( const A : Extended) : Integer;
7294:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7295:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7296:  Function WideCharToInt( const A : WideChar) : Integer
7297:  Function CharToInt( const A : Char) : Integer
7298:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7299:  Function IntToWideChar( const A : Integer) : WideChar
7300:  Function IntToChar( const A : Integer) : Char
7301:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7302:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7303:  Function IsHexChar( const Ch : Char) : Boolean
7304:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7305:  Function HexWideCharToInt( const A : WideChar) : Integer
7306:  Function HexCharToInt( const A : Char) : Integer
7307:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7308:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7309:  Function IntToUpperHexChar( const A : Integer) : Char
7310:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7311:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7312:  Function IntToLowerHexChar( const A : Integer) : Char
7313:  Function IntToStringA( const A : Int64) : AnsiString
7314:  Function IntToStringW( const A : Int64) : WideString
7315:  Function IntToString( const A : Int64) : String
7316:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7317:  Function UIntToStringW( const A : NativeUInt) : WideString
7318:  Function UIntToString( const A : NativeUInt) : String
7319:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7320:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7321:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7322:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7323:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7324:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7325:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7326:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7327:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7328:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
```

```
7329:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7330:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7331:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7332:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7333:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7334:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7335:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7336:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7337:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7338:  Function StringToInt64A( const S : AnsiString) : Int64
7339:  Function StringToInt64W( const S : WideString) : Int64
7340:  Function StringToInt64( const S : String) : Int64
7341:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7342:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7343:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7344:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7345:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7346:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7347:  Function StringToIntA( const S : AnsiString) : Integer
7348:  Function StringToIntW( const S : WideString) : Integer
7349:  Function StringToInt( const S : String) : Integer
7350:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7351:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7352:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7353:  Function StringToLongWordA( const S : AnsiString) : LongWord
7354:  Function StringToLongWordW( const S : WideString) : LongWord
7355:  Function StringToLongWord( const S : String) : LongWord
7356:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7357:  Function HexToUIntW( const S : WideString) : NativeUInt
7358:  Function HexToUInt( const S : String) : NativeUInt
7359:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7360:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7361:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7362:  Function HexToLongWordA( const S : AnsiString) : LongWord
7363:  Function HexToLongWordW( const S : WideString) : LongWord
7364:  Function HexToLongWord( const S : String) : LongWord
7365:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7366:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7367:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7368:  Function OctToLongWordA( const S : AnsiString) : LongWord
7369:  Function OctToLongWordW( const S : WideString) : LongWord
7370:  Function OctToLongWord( const S : String) : LongWord
7371:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7372:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7373:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7374:  Function BinToLongWordA( const S : AnsiString) : LongWord
7375:  Function BinToLongWordW( const S : WideString) : LongWord
7376:  Function BinToLongWord( const S : String) : LongWord
7377:  Function FloatToStringA( const A : Extended) : AnsiString
7378:  Function FloatToStringW( const A : Extended) : WideString
7379:  Function FloatToString( const A : Extended) : String
7380:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7381:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7382:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7383:  Function StringToFloatA( const A : AnsiString) : Extended
7384:  Function StringToFloatW( const A : WideString) : Extended
7385:  Function StringToFloat( const A : String) : Extended
7386:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7387:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7388:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7389:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
     PadChar: AnsiChar) : AnsiString
7390:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7391:  unit uPSI_cFundamentUtils;
7392:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7393:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7394:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7395:  Const('CCHARSET','Stringb64_XXEncode);
7396:  Const('CHEXSET','String'0123456789ABCDEF
7397:  Const('HEXDIGITS','String'0123456789ABCDEF
7398:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7399:  Const('DIGISET','String'0123456789
7400:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7401:  Const('DIGISET2','TCharset').SetSet('0123456789'
7402:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7403:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7404:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7405:  Const('NUMBERS','String'0123456789');
7406:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7407:  Function CharSetToStr( const C : CharSet) : AnsiString
7408:  Function StrToCharSet( const S : AnsiString) : CharSet
7409:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7410:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7411:  Function UUDecode( const S : AnsiString) : AnsiString
7412:  Function XXDecode( const S : AnsiString) : AnsiString
7413:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7414:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7415:  Function InterfaceToStrW( const I : IInterface) : WideString
7416:  Function InterfaceToStr( const I : IInterface) : String
```

```
7417:  Function ObjectClassName( const O : TObject) : String
7418:  Function ClassClassName( const C : TClass) : String
7419:  Function ObjectToStr( const O : TObject) : String
7420:  Function ObjectToString( const O : TObject) : String
7421:  Function CharSetToStr( const C : CharSet) : AnsiString
7422:  Function StrToCharSet( const S : AnsiString) : CharSet
7423:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7424:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7425:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7426:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7427:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7428:  Const('Bytes1KB','LongInt'( 1024);
7429:   SIRegister_IInterface(CL);
7430:  Procedure SelfTestCFundamentUtils
7431:
7432: Function CreateSchedule : IJclSchedule
7433: Function NullStamp : TTimeStamp
7434: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7435: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7436: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7437:
7438: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7439: begin
7440:  AddTypeS('TFunc', 'function(X : Float) : Float;
7441: Function InitGraphics( Width, Height : Integer) : Boolean
7442: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7443: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7444: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7445: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7446: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7447: Procedure SetGraphTitle( Title : String)
7448: Procedure SetOxTitle( Title : String)
7449: Procedure SetOyTitle( Title : String)
7450: Function GetGraphTitle : String
7451: Function GetOxTitle : String
7452: Function GetOyTitle : String
7453: Procedure PlotOxAxis( Canvas : TCanvas)
7454: Procedure PlotOyAxis( Canvas : TCanvas)
7455: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7456: Procedure WriteGraphTitle( Canvas : TCanvas)
7457: Function SetMaxCurv( NCurv : Byte) : Boolean
7458: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7459: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7460: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7461: Procedure SetCurvStep( CurvIndex, Step : Integer)
7462: Function GetMaxCurv : Byte
7463: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7464: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7465: Function GetCurvLegend( CurvIndex : Integer) : String
7466: Function GetCurvStep( CurvIndex : Integer) : Integer
7467: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7468: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7469: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7470: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7471: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7472: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7473: Function Xpixel( X : Float) : Integer
7474: Function Ypixel( Y : Float) : Integer
7475: Function Xuser( X : Integer) : Float
7476: Function Yuser( Y : Integer) : Float
7477: end;
7478:
7479: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7480: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7481: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7482: Procedure FFT_Integer_Cleanup
7483: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7484: //unit uPSI_JclStreams;
7485: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7486: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7487: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7488: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7489:
7490: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7491: begin
7492:  FindClass('TOBJECT'),'EInvalidDest
7493:  FindClass('TOBJECT'),'EFCantMove
7494:  Procedure fmxCopyFile( const FileName, DestName : string)
7495:  Procedure fmxMoveFile( const FileName, DestName : string)
7496:  Function fmxGetFileSize( const FileName : string) : LongInt
7497:  Function fmxFileDateTime( const FileName : string) : TDateTime
7498:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7499:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7500: end;
7501:
7502: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
```

```
7503: begin
7504:  SIRegister_IFindFileIterator(CL);
7505:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7506: end;
7507:
7508: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7509: begin
7510:  Function SkipWhite( cp : PChar) : PChar
7511:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7512:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7513:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7514: end;
7515:
7516: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7517: begin
7518:   SIRegister_TStringHashMapTraits(CL);
7519:  Function CaseSensitiveTraits : TStringHashMapTraits
7520:  Function CaseInsensitiveTraits : TStringHashMapTraits
7521:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7522:    +'e; Right : PHashNode; end
7523:   //PHashArray', '^THashArray // will not work
7524:   SIRegister_TStringHashMap(CL);
7525:  THashValue', 'Cardinal
7526:  Function StrHash( const s : string) : THashValue
7527:  Function TextHash( const s : string) : THashValue
7528:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7529:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7530:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7531:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7532:   SIRegister_TCaseSensitiveTraits(CL);
7533:   SIRegister_TCaseInsensitiveTraits(CL);
7534:
7535:
7536: //************************************************************************unit uPSI_umath;
7537:  Function uExpo( X : Float) : Float
7538:  Function uExp2( X : Float) : Float
7539:  Function uExp10( X : Float) : Float
7540:  Function uLog( X : Float) : Float
7541:  Function uLog2( X : Float) : Float
7542:  Function uLog10( X : Float) : Float
7543:  Function uLogA( X, A : Float) : Float
7544:  Function uIntPower( X : Float; N : Integer): Float
7545:  Function uPower( X, Y : Float) : Float
7546:  Function SgnGamma( X : Float) : Integer
7547:  Function Stirling( X : Float) : Float
7548:  Function StirLog( X : Float) : Float
7549:  Function Gamma( X : Float) : Float
7550:  Function LnGamma( X : Float) : Float
7551:  Function DiGamma( X : Float) : Float
7552:  Function TriGamma( X : Float) : Float
7553:  Function IGamma( X : Float) : Float
7554:  Function JGamma( X : Float) : Float
7555:  Function InvGamma( X : Float) : Float
7556:  Function Erf( X : Float) : Float
7557:  Function Erfc( X : Float) : Float
7558: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7559: { Correlation coefficient between samples X and Y }
7560: function DBeta(A, B, X : Float) : Float;
7561: { Density of Beta distribution with parameters A and B }
7562: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7563: Function Beta(X, Y : Float) : Float
7564: Function Binomial( N, K : Integer) : Float
7565: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7566: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7567: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7568: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7569: Function DNorm( X : Float) : Float
7570:
7571: function DGamma(A, B, X : Float) : Float;
7572: { Density of Gamma distribution with parameters A and B }
7573: function DKhi2(Nu : Integer; X : Float) : Float;
7574: { Density of Khi-2 distribution with Nu d.o.f. }
7575: function DStudent(Nu : Integer; X : Float) : Float;
7576: { Density of Student distribution with Nu d.o.f. }
7577: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7578: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7579: function IBeta(A, B, X : Float) : Float;
7580: { Incomplete Beta function}
7581: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7582:
7583: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7584: begin
7585:  Procedure SetOptAlgo( Algo : TOptAlgo)
7586: procedure SetOptAlgo(Algo : TOptAlgo);
7587: { ----------------------------------------------------------------
7588:   Sets the optimization algorithm according to Algo, which must be
7589:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7590:
7591:  Function GetOptAlgo : TOptAlgo
```

```
7592:  Procedure SetMaxParam( N : Byte)
7593:  Function GetMaxParam : Byte
7594:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7595:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7596:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7597:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
      Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7598:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
      MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7599:  Procedure SetMCFile( FileName : String)
7600:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7601:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
      LastPar:Integer;V:TMatrix);
7602: end;
7603:
7604: (*----------------------------------------------------------------------------*)
7605: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7606: begin
7607:  Procedure SaveSimplex( FileName : string)
7608:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7609: end;
7610: (*----------------------------------------------------------------------------*)
7611: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7612: begin
7613:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7614:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7615: end;
7616:
7617: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7618: begin
7619:  Function LTrim( S : String) : String
7620:  Function RTrim( S : String) : String
7621:  Function uTrim( S : String) : String
7622:  Function StrChar( N : Byte; C : Char) : String
7623:  Function RFill( S : String; L : Byte) : String
7624:  Function LFill( S : String; L : Byte) : String
7625:  Function CFill( S : String; L : Byte) : String
7626:  Function Replace( S : String; C1, C2 : Char) : String
7627:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7628:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7629:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7630:  Function FloatStr( X : Float) : String
7631:  Function IntStr( N : LongInt) : String
7632:  Function uCompStr( Z : Complex) : String
7633: end;
7634:
7635: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7636: begin
7637:  Function uSinh( X : Float) : Float
7638:  Function uCosh( X : Float) : Float
7639:  Function uTanh( X : Float) : Float
7640:  Function uArcSinh( X : Float) : Float
7641:  Function uArcCosh( X : Float) : Float
7642:  Function ArcTanh( X : Float) : Float
7643:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7644: end;
7645:
7646: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7647: begin
7648: type RNG_Type =
7649:   (RNG_MWC,      { Multiply-With-Carry }
7650:    RNG_MT,       { Mersenne Twister }
7651:    RNG_UVAG);    { Universal Virtual Array Generator }
7652:  Procedure SetRNG( RNG : RNG_Type)
7653:  Procedure InitGen( Seed : RNG_IntType)
7654:  Procedure SRand( Seed : RNG_IntType)
7655:  Function IRanGen : RNG_IntType
7656:  Function IRanGen31 : RNG_IntType
7657:  Function RanGen1 : Float
7658:  Function RanGen2 : Float
7659:  Function RanGen3 : Float
7660:  Function RanGen53 : Float
7661: end;
7662:
7663: //  Optimization by Simulated Annealing
7664: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7665: begin
7666:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7667:  Procedure SA_CreateLogFile( FileName : String)
7668:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7669: end;
7670:
7671: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7672: begin
7673:  Procedure InitUVAGbyString( KeyPhrase : string)
7674:  Procedure InitUVAG( Seed : RNG_IntType)
7675:  Function IRanUVAG : RNG_IntType
7676: end;
7677:
```

```
7678: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7679: begin
7680:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7681:  Procedure GA_CreateLogFile( LogFileName : String)
7682:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7683: end;
7684:
7685:  TVector', 'array of Float
7686: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7687: begin
7688:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7689:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7690: end;
7691:
7692: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7693: begin
7694:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7695:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7696: end;
7697:
7698: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7699: begin
7700:   FT_Result', 'Integer
7701:   //TDWordptr', '^DWord // will not work
7702:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7703:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7704:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7705:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7706:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7707:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7708:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7709:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7710:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7711:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7712:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7713:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7714:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7715:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7716:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7717:    yte; end
7718: end;
7719:
7720:
7721: //************************************ PaintFX*****************************
7722: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7723: begin
7724:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7725:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7726:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7727:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7728:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7729:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7730:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7731:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7732:     Procedure Turn( Src, Dst : TBitmap)
7733:     Procedure TurnRight( Src, Dst : TBitmap)
7734:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7735:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7736:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7737:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7738:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7739:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7740:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7741:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7742:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7743:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7744:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7745:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7746:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7747:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7748:     Procedure Emboss( var Bmp : TBitmap)
7749:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7750:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7751:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7752:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7753:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7754:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7755:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7756:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7757:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7758:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7759:     Procedure SemiOpaque( Src, Dst : TBitmap)
7760:     Procedure ShadowDownLeft( const Dst : TBitmap)
7761:     Procedure ShadowDownRight( const Dst : TBitmap)
7762:     Procedure ShadowUpLeft( const Dst : TBitmap)
7763:     Procedure ShadowUpRight( const Dst : TBitmap)
7764:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7765:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7766:     Procedure FlipRight( const Dst : TBitmap)
```

```
7767:     Procedure FlipDown( const Dst : TBitmap)
7768:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7769:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7770:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7771:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7772:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7773:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7774:     Procedure SmoothResize( var Src, Dst : TBitmap)
7775:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7776:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7777:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7778:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7779:     Procedure GrayScale( const Dst : TBitmap)
7780:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7781:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7782:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7783:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7784:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7785:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7786:     Procedure AntiAlias( const Dst : TBitmap)
7787:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7788:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7789:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7790:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7791:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7792:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7793:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7794:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7795:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7796:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7797:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7798:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7799:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7800:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7801:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7802:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7803:     Procedure Invert( Src : TBitmap)
7804:     Procedure MirrorRight( Src : TBitmap)
7805:     Procedure MirrorDown( Src : TBitmap)
7806:   end;
7807: end;
7808:
7809: (*----------------------------------------------------------------------------*)
7810: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7811: begin
7812:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7813:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7814:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7815:   SIRegister_TJvPaintFX(CL);
7816:  Function SplineFilter( Value : Single) : Single
7817:  Function BellFilter( Value : Single) : Single
7818:  Function TriangleFilter( Value : Single) : Single
7819:  Function BoxFilter( Value : Single) : Single
7820:  Function HermiteFilter( Value : Single) : Single
7821:  Function Lanczos3Filter( Value : Single) : Single
7822:  Function MitchellFilter( Value : Single) : Single
7823: end;
7824:
7825:
7826: (*----------------------------------------------------------------------------*)
7827: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7828: begin
7829:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7830:  TeeMsg_DefaultSeriesName','String 'Series
7831:  TeeMsg_DefaultToolName','String 'ChartTool
7832:  ChartComponentPalette','String 'TeeChart
7833:  TeeMaxLegendColumns',LongInt'( 2);
7834:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7835:  TeeTitleFootDistance,LongInt( 5);
7836:   SIRegister_TCustomChartWall(CL);
7837:   SIRegister_TChartWall(CL);
7838:   SIRegister_TChartLegendGradient(CL);
7839:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7840:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7841:   FindClass('TOBJECT'),'LegendException
7842:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7843:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7844:   FindClass('TOBJECT'),'TCustomChartLegend
7845:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7846:   TLegendSymbolPosition', '( spLeft, spRight )
7847:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7848:   TSymbolCalcHeight', 'Function  : Integer
7849:   SIRegister_TLegendSymbol(CL);
7850:   SIRegister_TTeeCustomShapePosition(CL);
7851:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7852:   SIRegister_TLegendTitle(CL);
7853:   SIRegister_TLegendItem(CL);
7854:   SIRegister_TLegendItems(CL);
7855:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
```

```
7856:   FindClass('TOBJECT'),'TCustomChart
7857:   SIRegister_TCustomChartLegend(CL);
7858:   SIRegister_TChartLegend(CL);
7859:   SIRegister_TChartTitle(CL);
7860:   SIRegister_TChartFootTitle(CL);
7861:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7862:    +'eButton; Shift : TShiftState; X, Y : Integer)
7863:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7864:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7865:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7866:    +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7867:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7868:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7869:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7870:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7871:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7872:    +'toMax : Boolean; Min : Double; Max : Double; end
7873:   TAllAxisSavedScales', 'array of TAxisSavedScales
7874:   SIRegister_TChartBackWall(CL);
7875:   SIRegister_TChartRightWall(CL);
7876:   SIRegister_TChartBottomWall(CL);
7877:   SIRegister_TChartLeftWall(CL);
7878:   SIRegister_TChartWalls(CL);
7879:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7880:   SIRegister_TCustomChart(CL);
7881:   SIRegister_TChart(CL);
7882:   SIRegister_TTeeSeriesTypes(CL);
7883:   SIRegister_TTeeToolTypes(CL);
7884:   SIRegister_TTeeDragObject(CL);
7885:   SIRegister_TColorPalettes(CL);
7886:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7887:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7888:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7889:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7890:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7891:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7892:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7893:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7894:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7895:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7896:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7897:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7898:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7899:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7900:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7901:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7902:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7903:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7904:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7905:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7906:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
      R:TRect;ReferenceChart:TCustomChart);
7907:   SIRegister_TChartTheme(CL);
7908:   //TChartThemeClass', 'class of TChartTheme
7909:   //TCanvasClass', 'class of TCanvas3D
7910:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7911:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7912:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7913:  Procedure ShowMessageUser( const S : String)
7914:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7915:  Function HasLabels( ASeries : TChartSeries) : Boolean
7916:  Function HasColors( ASeries : TChartSeries) : Boolean
7917:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7918:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7919: end;
7920:
7921:
7922: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7923: begin
7924:  //'TeeFormBorderStyle',' bsNone);
7925:   SIRegister_TMetafile(CL);
7926:  'TeeDefVerticalMargin','LongInt'( 4);
7927:  'TeeDefHorizMargin','LongInt'( 3);
7928:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7929:  'TeeMsg_TeeHand','String 'crTeeHand
7930:  'TeeNormalPrintDetail','LongInt'( 0);
7931:  'TeeHighPrintDetail','LongInt'( - 100);
7932:  'TeeDefault_PrintMargin','LongInt'( 15);
7933:  'MaxDefaultColors','LongInt'( 19);
7934:  'TeeTabDelimiter','Char #9);
7935:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7936:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7937:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7938:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7939:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
```

```
7940:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7941:   SIRegister_TCustomPanelNoCaption(CL);
7942:   FindClass('TOBJECT'),'TCustomTeePanel
7943:   SIRegister_TZoomPanning(CL);
7944:   SIRegister_TTeeEvent(CL);
7945:   //SIRegister_TTeeEventListeners(CL);
7946:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7947:   SIRegister_TTeeMouseEvent(CL);
7948:   SIRegister_TCustomTeePanel(CL);
7949:   //TChartGradient', 'TTeeGradient
7950:   //TChartGradientClass', 'class of TChartGradient
7951:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7952:   SIRegister_TTeeZoomPen(CL);
7953:   SIRegister_TTeeZoomBrush(CL);
7954:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7955:   SIRegister_TTeeZoom(CL);
7956:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7957:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7958:   SIRegister_TBackImage(CL);
7959:   SIRegister_TCustomTeePanelExtended(CL);
7960:   //TChartBrushClass', 'class of TChartBrush
7961:   SIRegister_TTeeCustomShapeBrushPen(CL);
7962:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7963:   TTextFormat', '( ttfNormal, ttfHtml )
7964:   SIRegister_TTeeCustomShape(CL);
7965:   SIRegister_TTeeShape(CL);
7966:   SIRegister_TTeeExportData(CL);
7967:  Function TeeStr( const Num : Integer) : String
7968:  Function DateTimeDefaultFormat( const AStep : Double) : String
7969:  Function TEEDaysInMonth( Year, Month : Word) : Word
7970:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7971:  Function NextDateTimeStep( const AStep : Double) : Double
7972:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7973:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7974:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7975:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7976:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7977:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7978:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7979:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7980:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7981:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7982:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7983:  Function DelphiToLocalFormat( const Format : String) : String
7984:  Function LocalToDelphiFormat( const Format : String) : String
7985:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
7986:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7987:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7988:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
7989:   TTeeSortSwap', 'Procedure ( a, b : Integer)
7990:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
7991:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7992:  Function TeeExtractField( St : String; Index : Integer) : String;
7993:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7994:  Function TeeNumFields( St : String) : Integer;
7995:  Function TeeNumFields1( const St, Separator : String) : Integer;
7996:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7997:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7998:  // TColorArray', 'array of TColor
7999:  Function GetDefaultColor( const Index : Integer) : TColor
8000:  Procedure SetDefaultColorPalette;
8001:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
8002:  'TeeCheckBoxSize','LongInt'( 11);
8003:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8004:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
8005:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8006:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8007:  Procedure TeeTranslateControl( AControl : TControl);
8008:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
8009:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8010:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8011:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8012:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8013:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8014:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8015:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8016:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8017:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8018:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8019:  Procedure TeeSaveStringOption( const AKey, Value : String)
8020:  Function TeeDefaultXMLEncoding : String
8021:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8022:   TeeWindowHandle', 'Integer
8023:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
8024:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8025:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8026: end;
8027:
```

```
8028:
8029: using mXBDEUtils
8030: ********************************************************************************
8031:  Procedure SetAlias( aAlias, aDirectory : String)
8032:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
8033:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
8034:  Procedure SetBDE( aPath, aNode, aValue : String)
8035:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8036:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8037:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8038:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8039:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8040:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8041:
8042:
8043: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8044: begin
8045: AddClassN(FindClass('TOBJECT'),'EDateTime
8046: Function DatePart( const D : TDateTime) : Integer
8047: Function TimePart( const D : TDateTime) : Double
8048: Function Century( const D : TDateTime) : Word
8049: Function Year( const D : TDateTime) : Word
8050: Function Month( const D : TDateTime) : Word
8051: Function Day( const D : TDateTime) : Word
8052: Function Hour( const D : TDateTime) : Word
8053: Function Minute( const D : TDateTime) : Word
8054: Function Second( const D : TDateTime) : Word
8055: Function Millisecond( const D : TDateTime) : Word
8056: ('OneDay','Extended').setExtended( 1.0);
8057: ('OneHour','Extended').SetExtended( OneDay / 24);
8058: ('OneMinute','Extended').SetExtended( OneHour / 60);
8059: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8060: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8061: ('OneWeek','Extended').SetExtended( OneDay * 7);
8062: ('HoursPerDay','Extended').SetExtended( 24);
8063: ('MinutesPerHour','Extended').SetExtended( 60);
8064: ('SecondsPerMinute','Extended').SetExtended( 60);
8065: Procedure SetYear( var D : TDateTime; const Year : Word)
8066: Procedure SetMonth( var D : TDateTime; const Month : Word)
8067: Procedure SetDay( var D : TDateTime; const Day : Word)
8068: Procedure SetHour( var D : TDateTime; const Hour : Word)
8069: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8070: Procedure SetSecond( var D : TDateTime; const Second : Word)
8071: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8072: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8073: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8074: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8075: Function IsAM( const D : TDateTime) : Boolean
8076: Function IsPM( const D : TDateTime) : Boolean
8077: Function IsMidnight( const D : TDateTime) : Boolean
8078: Function IsNoon( const D : TDateTime) : Boolean
8079: Function IsSunday( const D : TDateTime) : Boolean
8080: Function IsMonday( const D : TDateTime) : Boolean
8081: Function IsTuesday( const D : TDateTime) : Boolean
8082: Function IsWedneday( const D : TDateTime) : Boolean
8083: Function IsThursday( const D : TDateTime) : Boolean
8084: Function IsFriday( const D : TDateTime) : Boolean
8085: Function IsSaturday( const D : TDateTime) : Boolean
8086: Function IsWeekend( const D : TDateTime) : Boolean
8087: Function Noon( const D : TDateTime) : TDateTime
8088: Function Midnight( const D : TDateTime) : TDateTime
8089: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8090: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8091: Function NextWorkday( const D : TDateTime) : TDateTime
8092: Function PreviousWorkday( const D : TDateTime) : TDateTime
8093: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8094: Function LastDayOfYear( const D : TDateTime) : TDateTime
8095: Function EasterSunday( const Year : Word) : TDateTime
8096: Function GoodFriday( const Year : Word) : TDateTime
8097: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8098: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8099: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8100: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8101: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8102: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8103: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8104: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8105: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8106: Function DayOfYear( const D : TDateTime) : Integer
8107: Function DaysInMonth( const Ye, Mo : Word) : Integer
8108: Function DaysInMonth( const D : TDateTime) : Integer
8109: Function DaysInYear( const Ye : Word) : Integer
8110: Function DaysInYearDate( const D : TDateTime) : Integer
8111: Function WeekNumber( const D : TDateTime) : Integer
8112: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8113: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8114: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8115: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
```

```
8116: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8117: Function DiffHours( const D1, D2 : TDateTime) : Integer
8118: Function DiffDays( const D1, D2 : TDateTime) : Integer
8119: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8120: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8121: Function DiffYears( const D1, D2 : TDateTime) : Integer
8122: Function GMTBias : Integer
8123: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8124: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8125: Function NowAsGMTTime : TDateTime
8126: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8127: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8128: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8129: Function DateTimeToANSI( const D : TDateTime) : Integer
8130: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8131: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8132: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8133: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8134: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8135: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8136: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8137: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8138: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8139: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8140: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8141: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8142: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8143: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8144: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8145: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8146: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8147: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8148: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8149: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8150: Function EnglishShortMonthA( const S : AnsiString) : Integer
8151: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8152: Function EnglishLongMonthA( const S : AnsiString) : Integer
8153: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8154: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8155: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8156: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8157: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8158: Function RFCMonthA( const S : AnsiString) : Word
8159: Function RFCMonthU( const S : UnicodeString) : Word
8160: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8161: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8162: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8163: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8164: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8165: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8166: Function NowAsRFCDateTimeA : AnsiString
8167: Function NowAsRFCDateTimeU : UnicodeString
8168: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8169: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8170: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8171: Function TimePeriodStr( const D : TDateTime) : AnsiString
8172: Procedure SelfTest
8173: end;
8174: //*****************************************CFileUtils
8175: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8176: Function PathHasDriveLetter( const Path : String) : Boolean
8177: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8178: Function PathIsDriveLetter( const Path : String) : Boolean
8179: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8180: Function PathIsDriveRoot( const Path : String) : Boolean
8181: Function PathIsRootA( const Path : AnsiString) : Boolean
8182: Function PathIsRoot( const Path : String) : Boolean
8183: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8184: Function PathIsUNCPath( const Path : String) : Boolean
8185: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8186: Function PathIsAbsolute( const Path : String) : Boolean
8187: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8188: Function PathIsDirectory( const Path : String) : Boolean
8189: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8190: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8191: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8192: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8193: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8194: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8195: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8196: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8197: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8198: Function PathCanonical( const Path : String; const PathSep : Char) : String
8199: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8200: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8201: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8202: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8203: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8204: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
```

```
8205: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8206: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8207: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8208: Function FileNameValid( const FileName : String) : String
8209: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8210: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8211: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8212: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8213: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8214: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8215: Procedure CCopyFile( const FileName, DestName : String)
8216: Procedure CMoveFile( const FileName, DestName : String)
8217: Function CDeleteFiles( const FileMask : String) : Boolean
8218: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8219: Procedure FileCloseEx( const FileHandle : TFileHandle)
8220: Function FileExistsA( const FileName : AnsiString) : Boolean
8221:  Function CFileExists( const FileName : String) : Boolean
8222:  Function CFileGetSize( const FileName : String) : Int64
8223: Function FileGetDateTime( const FileName : String) : TDateTime
8224: Function FileGetDateTime2( const FileName : String) : TDateTime
8225: Function FileIsReadOnly( const FileName : String) : Boolean
8226: Procedure FileDeleteEx( const FileName : String)
8227: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8228: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
      : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8229: Function DirectoryEntryExists( const Name : String) : Boolean
8230: Function DirectoryEntrySize( const Name : String) : Int64
8231:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8232: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8233: Procedure CDirectoryCreate( const DirectoryName : String)
8234: Function GetFirstFileNameMatching( const FileMask : String) : String
8235: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8236: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8237: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8238:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8239:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8240: Function DriveIsValid( const Drive : Char) : Boolean
8241: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8242: Function DriveFreeSpace( const Path : AnsiString) : Int64
8243:
8244: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8245: begin
8246:  AddClassN(FindClass('TOBJECT'),'ETimers
8247:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8248: Function TickDelta( const D1, D2 : LongWord) : Integer
8249: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8250:   AddTypeS('THPTimer', 'Int64
8251: Procedure StartTimer( var Timer : THPTimer)
8252: Procedure StopTimer( var Timer : THPTimer)
8253: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8254: Procedure InitStoppedTimer( var Timer : THPTimer)
8255: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8256: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8257: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8258: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8259: Function GetHighPrecisionFrequency : Int64
8260: Function GetHighPrecisionTimerOverhead : Int64
8261: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8262: Procedure SelfTestCTimer
8263: end;
8264:
8265: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8266: begin
8267:  Function RandomSeed : LongWord
8268:  Procedure AddEntropy( const Value : LongWord)
8269:  Function RandomUniform : LongWord;
8270:  Function RandomUniform1( const N : Integer) : Integer;
8271:  Function RandomBoolean : Boolean
8272:  Function RandomByte : Byte
8273:  Function RandomByteNonZero : Byte
8274:  Function RandomWord : Word
8275:  Function RandomInt64 : Int64;
8276:  Function RandomInt641( const N : Int64) : Int64;
8277:  Function RandomHex( const Digits : Integer) : String
8278:  Function RandomFloat : Extended
8279:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8280:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8281:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8282:  Function mwcRandomLongWord : LongWord
8283:  Function urnRandomLongWord : LongWord
8284:  Function moaRandomFloat : Extended
8285:  Function mwcRandomFloat : Extended
8286:  Function RandomNormalF : Extended
8287:  Procedure SelfTestCRandom
8288: end;
8289:
8290: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8291: begin
8292:  // PIntArray', '^TIntArray // will not work
```

```
8293:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8294:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8295:  Function synMax( x, y : integer) : integer
8296:  Function synMin( x, y : integer) : integer
8297:  Function synMinMax( x, mi, ma : integer) : integer
8298:  Procedure synSwapInt( var l, r : integer)
8299:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8300:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8301:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8302:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8303:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8304:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8305:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8306:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8307:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8308:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8309:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8310:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8311:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8312:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8313:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8314:  ('C3_NONSPACING','LongInt'( 1);
8315:  'C3_DIACRITIC','LongInt'( 2);
8316:  'C3_VOWELMARK','LongInt'( 4);
8317:  ('C3_SYMBOL','LongInt'( 8);
8318:  ('C3_KATAKANA','LongWord( $0010);
8319:  ('C3_HIRAGANA','LongWord( $0020);
8320:  ('C3_HALFWIDTH','LongWord( $0040);
8321:  ('C3_FULLWIDTH','LongWord( $0080);
8322:  ('C3_IDEOGRAPH','LongWord( $0100);
8323:  ('C3_KASHIDA','LongWord( $0200);
8324:  ('C3_LEXICAL','LongWord( $0400);
8325:  ('C3_ALPHA','LongWord( $8000);
8326:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8327:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8328:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8329:  Function synIsStringType( Value : Word) : TStringType
8330:  Function synGetEOL( Line : PChar) : PChar
8331:  Function synEncodeString( s : string) : string
8332:  Function synDecodeString( s : string) : string
8333:  Procedure synFreeAndNil( var Obj: TObject)
8334:  Procedure synAssert( Expr : Boolean)
8335:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8336:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8337:   TReplaceFlags', 'set of TReplaceFlag )
8338:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8339:  Function synGetRValue( RGBValue : TColor) : byte
8340:  Function synGetGValue( RGBValue : TColor) : byte
8341:  Function synGetBValue( RGBValue : TColor) : byte
8342:  Function synRGB( r, g, b : Byte) : Cardinal
8343: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8344:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8345:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8346:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8347:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8348: end;
8349:
8350:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8351:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8352:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8353:
8354: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8355: begin
8356:  Function STimeZoneBias : integer
8357:  Function TimeZone : string
8358:  Function Rfc822DateTime( t : TDateTime) : string
8359:  Function CDateTime( t : TDateTime) : string
8360:  Function SimpleDateTime( t : TDateTime) : string
8361:  Function AnsiCDateTime( t : TDateTime) : string
8362:  Function GetMonthNumber( Value : String) : integer
8363:  Function GetTimeFromStr( Value : string) : TDateTime
8364:  Function GetDateMDYFromStr( Value : string) : TDateTime
8365:  Function DecodeRfcDateTime( Value : string) : TDateTime
8366:  Function GetUTTime : TDateTime
8367:  Function SetUTTime( Newdt : TDateTime) : Boolean
8368:  Function SGetTick : LongWord
8369:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8370:  Function CodeInt( Value : Word) : Ansistring
8371:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8372:  Function CodeLongInt( Value : LongInt) : Ansistring
8373:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8374:  Function DumpStr( const Buffer : Ansistring) : string
8375:  Function DumpExStr( const Buffer : Ansistring) : string
8376:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8377:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8378:  Function TrimSPLeft( const S : string) : string
8379:  Function TrimSPRight( const S : string) : string
```

```
8380:  Function TrimSP( const S : string) : string
8381:  Function SeparateLeft( const Value, Delimiter : string) : string
8382:  Function SeparateRight( const Value, Delimiter : string) : string
8383:  Function SGetParameter( const Value, Parameter : string) : string
8384:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8385:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8386:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8387:  Function GetEmailAddr( const Value : string) : string
8388:  Function GetEmailDesc( Value : string) : string
8389:  Function CStrToHex( const Value : Ansistring) : string
8390:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8391:  Function CBinToInt( const Value : string) : Integer
8392:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8393:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8394:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8395:  Function CRPos( const Sub, Value : String) : Integer
8396:  Function FetchBin( var Value : string; const Delimiter : string) : string
8397:  Function CFetch( var Value : string; const Delimiter : string) : string
8398:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8399:  Function IsBinaryString( const Value : AnsiString) : Boolean
8400:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8401:  Procedure StringsTrim( const value : TStrings)
8402:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8403:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8404:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8405:  Function CCountOfChar( const Value : string; aChr : char) : integer
8406:  Function UnquoteStr( const Value : string; Quote : Char) : string
8407:  Function QuoteStr( const Value : string; Quote : Char) : string
8408:  Procedure HeadersToList( const Value : TStrings)
8409:  Procedure ListToHeaders( const Value : TStrings)
8410:  Function SwapBytes( Value : integer) : integer
8411:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8412:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8413:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8414:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8415:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8416:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8417: end;
8418:
8419: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8420: begin
8421:  ('CrcBufSize','LongInt'( 2048);
8422:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8423:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8424:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8425:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8426:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8427:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8428:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8429:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8430:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8431:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8432:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8433:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8434:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8435:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8436:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8437: end;
8438:
8439: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8440: begin
8441:  function CreateOleObject(const ClassName: String): IDispatch;
8442:  function GetActiveOleObject(const ClassName: String): IDispatch;
8443:  function ProgIDToClassID(const ProgID: string): TGUID;
8444:  function ClassIDToProgID(const ClassID: TGUID): string;
8445:  function CreateClassID: string;
8446:  function CreateGUIDString: string;
8447:  function CreateGUIDID: string;
8448:  procedure OleError(ErrorCode: longint)
8449:  procedure OleCheck(Result: HResult);
8450: end;
8451:
8452:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8453:  Function xGetActiveOleObject( const ClassName : string) : Variant
8454:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8455:  Function DllCanUnloadNow : HResult
8456:  Function DllRegisterServer : HResult
8457:  Function DllUnregisterServer : HResult
8458:  Function VarFromInterface( Unknown : IUnknown) : Variant
8459:  Function VarToInterface( const V : Variant) : IDispatch
8460:  Function VarToAutoObject( const V : Variant) : TAutoObject
8461:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8462:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8463:  Procedure OleError( ErrorCode : HResult)
8464:  Procedure OleCheck( Result : HResult)
8465:  Function StringToClassID( const S : string) : TCLSID
8466:  Function ClassIDToString( const ClassID : TCLSID) : string
8467:  Function xProgIDToClassID( const ProgID : string) : TCLSID
```

```
8468:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8469:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8470:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8471:  Function xGUIDToString( const ClassID : TGUID) : string
8472:  Function xStringToGUID( const S : string) : TGUID
8473:  Function xGetModuleName( Module : HMODULE) : string
8474:  Function xAcquireExceptionObject : TObject
8475:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8476:  Function xUtf8Encode( const WS : WideString) : UTF8String
8477:  Function xUtf8Decode( const S : UTF8String) : WideString
8478:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8479:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8480:  Function XRTLHandleCOMException : HResult
8481:  Procedure XRTLCheckArgument( Flag : Boolean)
8482:  //Procedure XRTLCheckOutArgument( out Arg)
8483:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8484:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8485:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8486:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8487:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8488:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8489: function   XRTLDefaultCategoryManager: IUnknown;
8490: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8491: // ICatRegister helper functions
8492: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8493:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8494:                                        const CategoryManager: IUnknown = nil): HResult;
8495: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8496:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8497:                                        const CategoryManager: IUnknown = nil): HResult;
8498: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8499:                                        const CategoryManager: IUnknown = nil): HResult;
8500: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8501:                                          const CategoryManager: IUnknown = nil): HResult;
8502: // ICatInformation helper functions
8503: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8504:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8505:                                       const CategoryManager: IUnknown = nil): HResult;
8506: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8507:                                const CategoryManager: IUnknown = nil): HResult;
8508: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8509:                                     const CategoryManager: IUnknown = nil): HResult;
8510: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8511:                                      const CategoryManager: IUnknown = nil): HResult;
8512: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8513:                    const ADelete: Boolean = True): WideString;
8514: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8515: Function XRTLGetVariantAsString( const Value : Variant) : string
8516: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8517: Function XRTLGetTimeZones : TXRTLTimeZones
8518: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8519: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8520: Function GMTNow : TDateTime
8521: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8522: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8523: Procedure XRTLNotImplemented
8524: Procedure XRTLRaiseError( E : Exception)
8525: Procedure XRTLRaise( E : Exception)');
8526: Procedure XRaise( E : Exception)');
8527: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8528:
8529:
8530: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8531: begin
8532:   SIRegister_IXRTLValue(CL);
8533:   SIRegister_TXRTLValue(CL);
8534:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8535:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8536: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8537: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8538: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8539: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8540: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8541: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8542: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8543: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8544: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8545: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8546: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8547: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8548: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8549: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8550: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8551: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8552: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8553: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8554: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
```

```
8555: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8556: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8557: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8558: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8559: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8560: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8561: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8562: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8563:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8564: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8565: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8566: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8567: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8568: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8569: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8570: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8571: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8572: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8573: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8574: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8575: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8576: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8577: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8578: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8579: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8580: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8581: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8582: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8583: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8584: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8585: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8586: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8587: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8588: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8589: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8590: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8591: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8592: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8593: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8594: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8595: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8596: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8597: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8598: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8599: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8600: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8601: end;
8602:
8603: //*****************************unit uPSI_GR32;****************************************
8604:
8605:  Function Color32( WinColor : TColor) : TColor32;
8606:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8607:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8608:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8609:  Function WinColor( Color32 : TColor32) : TColor
8610:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8611:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8612:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8613:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8614:  Function RedComponent( Color32 : TColor32) : Integer
8615:  Function GreenComponent( Color32 : TColor32) : Integer
8616:  Function BlueComponent( Color32 : TColor32) : Integer
8617:  Function AlphaComponent( Color32 : TColor32) : Integer
8618:  Function Intensity( Color32 : TColor32) : Integer
8619:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8620:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8621:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8622:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8623:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8624:  Function WinPalette( const P : TPalette32) : HPALETTE
8625:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8626:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8627:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8628:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8629:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8630:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8631:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8632:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8633:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8634:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8635:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8636:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8637:  Function FixedRect1( const ARect : TRect) : TRect;
8638:  Function FixedRect2( const FR : TFloatRect) : TRect;
8639:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8640:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8641:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8642:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
```

```
8643:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8644:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8645:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8646:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8647:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8648:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8649:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8650:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8651:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8652:  Function IsRectEmpty( const R : TRect) : Boolean;
8653:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8654:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8655:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8656:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8657:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8658:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8659:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8660:  Function MessageBeep( uType : UINT) : BOOL
8661:  Function ShowCursor( bShow : BOOL) : Integer
8662:  Function SetCursorPos( X, Y : Integer) : BOOL
8663:  Function SetCursor( hCursor : HICON) : HCURSOR
8664:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8665:  //Function ClipCursor( lpRect : PRect) : BOOL
8666:  Function GetClipCursor( var lpRect : TRect) : BOOL
8667:  Function GetCursor : HCURSOR
8668:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8669:  Function GetCaretBlinkTime : UINT
8670:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8671:  Function DestroyCaret : BOOL
8672:  Function HideCaret( hWnd : HWND) : BOOL
8673:  Function ShowCaret( hWnd : HWND) : BOOL
8674:  Function SetCaretPos( X, Y : Integer) : BOOL
8675:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8676:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8677:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8678:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8679:  Function WindowFromPoint( Point : TPoint) : HWND
8680:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8681:
8682:
8683: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8684: begin
8685:  Function FixedFloor( A : TFixed) : Integer
8686:  Function FixedCeil( A : TFixed) : Integer
8687:  Function FixedMul( A, B : TFixed) : TFixed
8688:  Function FixedDiv( A, B : TFixed) : TFixed
8689:  Function OneOver( Value : TFixed) : TFixed
8690:  Function FixedRound( A : TFixed) : Integer
8691:  Function FixedSqr( Value : TFixed) : TFixed
8692:  Function FixedSqrtLP( Value : TFixed) : TFixed
8693:  Function FixedSqrtHP( Value : TFixed) : TFixed
8694:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8695:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8696:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8697:  Function GRHypot( const X, Y : TFloat) : TFloat;
8698:  Function Hypot1( const X, Y : Integer) : Integer;
8699:  Function FastSqrt( const Value : TFloat) : TFloat
8700:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8701:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8702:  Function FastInvSqrt( const Value : Single) : Single;
8703:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8704:  Function GRIsPowerOf2( Value : Integer) : Boolean
8705:  Function PrevPowerOf2( Value : Integer) : Integer
8706:  Function NextPowerOf2( Value : Integer) : Integer
8707:  Function Average( A, B : Integer) : Integer
8708:  Function GRSign( Value : Integer) : Integer
8709:  Function FloatMod( x, y : Double) : Double
8710: end;
8711:
8712: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8713: begin
8714:  Function Clamp( const Value : Integer) : Integer;
8715:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8716:  Function StackAlloc( Size : Integer) : Pointer
8717:  Procedure StackFree( P : Pointer)
8718:  Procedure Swap( var A, B : Pointer);
8719:  Procedure Swap1( var A, B : Integer);
8720:  Procedure Swap2( var A, B : TFixed);
8721:  Procedure Swap3( var A, B : TColor32);
8722:  Procedure TestSwap( var A, B : Integer);
8723:  Procedure TestSwap1( var A, B : TFixed);
8724:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8725:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8726:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8727:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8728:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8729:  Function GRMin( const A, B, C : Integer) : Integer;
8730:  Function GRMax( const A, B, C : Integer) : Integer;
8731:  Function Clamp( Value, Max : Integer) : Integer;
```

```
8732:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8733:  Function Wrap( Value, Max : Integer) : Integer;
8734:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8735:  Function Wrap3( Value, Max : Single) : Single;;
8736:  Function WrapPow2( Value, Max : Integer) : Integer;
8737:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8738:  Function Mirror( Value, Max : Integer) : Integer;
8739:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8740:  Function MirrorPow2( Value, Max : Integer) : Integer;
8741:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8742:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8743:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8744:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8745:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8746:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8747:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8748:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8749:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8750:  Function Div255( Value : Cardinal) : Cardinal
8751:  Function SAR_4( Value : Integer) : Integer
8752:  Function SAR_8( Value : Integer) : Integer
8753:  Function SAR_9( Value : Integer) : Integer
8754:  Function SAR_11( Value : Integer) : Integer
8755:  Function SAR_12( Value : Integer) : Integer
8756:  Function SAR_13( Value : Integer) : Integer
8757:  Function SAR_14( Value : Integer) : Integer
8758:  Function SAR_15( Value : Integer) : Integer
8759:  Function SAR_16( Value : Integer) : Integer
8760:  Function ColorSwap( WinColor : TColor) : TColor32
8761: end;
8762:
8763: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8764: begin
8765:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8766:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8767:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8768:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8769:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8770:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8771:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8772:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8773:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8774:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8775:  Function CreateBitmask( Components : TColor32Components) : TColor32
8776:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8777:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8778:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8779: end;
8780:
8781:
8782: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8783: begin
8784:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8785:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8786:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8787:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8788:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8789:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8790:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8791:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8792:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8793:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8794:    //+'tedRangeBuffer; MoreData : Boolean; end
8795:  Function NtfsSetSparse( const FileName : string) : Boolean
8796:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8797:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8798:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8799:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8800:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8801:  Function NtfsGetSparse( const FileName : string) : Boolean
8802:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8803:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8804:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8805:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8806:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8807:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8808:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8809:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8810:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8811:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8812:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8813:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8814:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8815:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
```

```
8816:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8817:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8818:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8819:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8820:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8821:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8822:   AddTypeS('TStreamIds', 'set of TStreamId
8823:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8824:    +': ___Pointer; StreamIds : TStreamIds; end
8825:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8826:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8827:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8828:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8829:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8830:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8831:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8832:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8833:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8834:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8835:  Function JclAppInstances : TJclAppInstances;
8836:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8837:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8838:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8839:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8840:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8841:
8842:
8843: (*---------------------------------------------------------------------------*)
8844: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8845: begin
8846:   FindClass('TOBJECT'),'EJclGraphicsError
8847:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8848:   TDynPointArray', 'array of TPoint
8849:   TDynDynPointArrayArray', 'array of TDynPointArray
8850:   TPointF', 'record X : Single; Y : Single; end
8851:   TDynPointArrayF', 'array of TPointF
8852:   TDrawMode2', '( dmOpaque, dmBlend )
8853:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8854:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8855:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8856:   TMatrix3d', 'record array[0..2,0..2] of extended end
8857:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8858:   TScanLine', 'array of Integer
8859:   TScanLines', 'array of TScanLine
8860:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8861:   TGradientDirection', '( gdVertical, gdHorizontal )
8862:   TPolyFillMode', '( fmAlternate, fmWinding )
8863:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8864:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8865:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8866:   SIRegister_TJclDesktopCanvas(CL);
8867:   FindClass('TOBJECT'),'TJclRegion
8868:   SIRegister_TJclRegionInfo(CL);
8869:   SIRegister_TJclRegion(CL);
8870:   SIRegister_TJclThreadPersistent(CL);
8871:   SIRegister_TJclCustomMap(CL);
8872:   SIRegister_TJclBitmap32(CL);
8873:   SIRegister_TJclByteMap(CL);
8874:   SIRegister_TJclTransformation(CL);
8875:   SIRegister_TJclLinearTransformation(CL);
8876:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8877:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8878:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8879:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8880:  Procedure BitmapToJPeg( const FileName : string)
8881:  Procedure JPegToBitmap( const FileName : string)
8882:  Function ExtractIconCount( const FileName : string) : Integer
8883:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8884:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8885:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8886:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8887:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8888:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8889:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8890:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8891:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8892:  Procedure ScreenShot1( bm : TBitmap);
8893:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8894:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8895:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8896:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8897:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8898:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
```

```
8899:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8900:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8901:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8902:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8903:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8904:  Procedure Invert( Dst, Src : TJclBitmap32)
8905:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8906:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8907:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8908:  Procedure SetGamma( Gamma : Single)
8909: end;
8910:
8911: (*----------------------------------------------------------------------------*)
8912: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8913: begin
8914:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8915:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8916:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8917:  Function LockedDec( var Target : Integer) : Integer
8918:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8919:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8920:  Function LockedExchangeDec( var Target : Integer) : Integer
8921:  Function LockedExchangeInc( var Target : Integer) : Integer
8922:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8923:  Function LockedInc( var Target : Integer) : Integer
8924:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8925:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8926:   SIRegister_TJclDispatcherObject(CL);
8927:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8928:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8929:   SIRegister_TJclCriticalSection(CL);
8930:   SIRegister_TJclCriticalSectionEx(CL);
8931:   SIRegister_TJclEvent(CL);
8932:   SIRegister_TJclWaitableTimer(CL);
8933:   SIRegister_TJclSemaphore(CL);
8934:   SIRegister_TJclMutex(CL);
8935:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8936:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8937:   SIRegister_TJclOptex(CL);
8938:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8939:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8940:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8941:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8942:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8943:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8944:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8945:   PMeteredSection', '^TMeteredSection // will not work
8946:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8947:   SIRegister_TJclMeteredSection(CL);
8948:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8949:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8950:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8951:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8952:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8953:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8954:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8955:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8956:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8957:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8958:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8959:   FindClass('TOBJECT'),'EJclCriticalSectionError
8960:   FindClass('TOBJECT'),'EJclEventError
8961:   FindClass('TOBJECT'),'EJclWaitableTimerError
8962:   FindClass('TOBJECT'),'EJclSemaphoreError
8963:   FindClass('TOBJECT'),'EJclMutexError
8964:   FindClass('TOBJECT'),'EJclMeteredSectionError
8965: end;
8966:
8967:
8968: //*************************unit uPSI_mORMotReport;
8969: Procedure SetCurrentPrinterAsDefault
8970: Function CurrentPrinterName : string
8971: Function mCurrentPrinterPaperSize : string
8972: Procedure UseDefaultPrinter
8973:
8974: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
8975: begin
8976:   with FindClass('TOBJECT'), 'TStream') do begin
8977:     IsAbstract := True;
8978:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8979:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8980:     function Read(Buffer:String;Count:LongInt):LongInt
8981:     function Write(Buffer:String;Count:LongInt):LongInt
8982:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
8983:     function WriteString(Buffer:String;Count:LongInt):LongInt
8984:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8985:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
```

```
8986:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8987:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8988:
8989:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8990:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8991:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8992:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8993:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8994:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8995:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8996:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8997:
8998:     function Seek(Offset:LongInt;Origin:Word):LongInt
8999:     procedure ReadBuffer(Buffer:String;Count:LongInt)
9000:     procedure WriteBuffer(Buffer:String;Count:LongInt)
9001:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
9002:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
9003:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
9004:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
9005:
9006:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9007:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9008:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9009:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9010:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9011:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9012:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9013:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9014:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9015:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9016:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9017:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9018:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9019:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9020:
9021:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9022:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9023:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
9024:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
9025:     //READBUFFERAC
9026:     function InstanceSize: Longint
9027:     Procedure FixupResourceHeader( FixupInfo : Integer)
9028:     Procedure ReadResHeader
9029:
9030:     {$IFDEF DELPHI4UP}
9031:     function CopyFrom(Source:TStream;Count:Int64):LongInt
9032:     {$ELSE}
9033:     function CopyFrom(Source:TStream;Count:Integer):LongInt
9034:     {$ENDIF}
9035:     RegisterProperty('Position', 'LongInt', iptrw);
9036:     RegisterProperty('Size', 'LongInt', iptrw);
9037:   end;
9038: end;
9039:
9040:
9041: { ************************************************************
9042:   Unit DMATH - Interface for DMATH.DLL
9043:   ************************************************************ }
9044: // see more docs/dmath_manual.pdf
9045:
9046: Function InitEval : Integer
9047: Procedure SetVariable( VarName : Char; Value : Float)
9048: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9049: Function Eval( ExpressionString : String) : Float
9050:
9051: unit dmath; //types are in built, others are external in DLL
9052: interface
9053: {$IFDEF DELPHI}
9054: uses
9055:   StdCtrls, Graphics;
9056: {$ENDIF}
9057: { ------------------------------------------------------------
9058:   Types and constants
9059:   ------------------------------------------------------------ }
9060: {$i types.inc}
9061: { ------------------------------------------------------------
9062:   Error handling
9063:   ------------------------------------------------------------ }
9064: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9065: { Sets the error code }
9066: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9067: { Sets error code and default function value }
9068: function MathErr : Integer; external 'dmath';
9069: { Returns the error code }
9070: { ------------------------------------------------------------
9071:   Dynamic arrays
9072:   ------------------------------------------------------------ }
9073: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9074: { Sets the auto-initialization of arrays }
```

```
9075: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9076: { Creates floating point vector V[0..Ub] }
9077: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9078: { Creates integer vector V[0..Ub] }
9079: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9080: { Creates complex vector V[0..Ub] }
9081: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9082: { Creates boolean vector V[0..Ub] }
9083: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9084: { Creates string vector V[0..Ub] }
9085: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9086: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9087: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9088: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9089: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9090: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9091: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9092: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9093: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9094: { Creates string matrix A[0..Ub1, 0..Ub2] }
9095: { ------------------------------------------------------------
9096:   Minimum, maximum, sign and exchange
9097:   ------------------------------------------------------------ }
9098: function FMin(X, Y : Float) : Float; external 'dmath';
9099: { Minimum of 2 reals }
9100: function FMax(X, Y : Float) : Float; external 'dmath';
9101: { Maximum of 2 reals }
9102: function IMin(X, Y : Integer) : Integer; external 'dmath';
9103: { Minimum of 2 integers }
9104: function IMax(X, Y : Integer) : Integer; external 'dmath';
9105: { Maximum of 2 integers }
9106: function Sgn(X : Float) : Integer; external 'dmath';
9107: { Sign (returns 1 if X = 0) }
9108: function Sgn0(X : Float) : Integer; external 'dmath';
9109: { Sign (returns 0 if X = 0) }
9110: function DSgn(A, B : Float) : Float; external 'dmath';
9111: { Sgn(B) * |A| }
9112: procedure FSwap(var X, Y : Float); external 'dmath';
9113: { Exchange 2 reals }
9114: procedure ISwap(var X, Y : Integer); external 'dmath';
9115: { Exchange 2 integers }
9116: { ------------------------------------------------------------
9117:   Rounding functions
9118:   ------------------------------------------------------------ }
9119: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9120: { Rounds X to N decimal places }
9121: function Ceil(X : Float) : Integer; external 'dmath';
9122: { Ceiling function }
9123: function Floor(X : Float) : Integer; external 'dmath';
9124: { Floor function }
9125: { ------------------------------------------------------------
9126:   Logarithms, exponentials and power
9127:   ------------------------------------------------------------ }
9128: function Expo(X : Float) : Float; external 'dmath';
9129: { Exponential }
9130: function Exp2(X : Float) : Float; external 'dmath';
9131: { 2^X }
9132: function Exp10(X : Float) : Float; external 'dmath';
9133: { 10^X }
9134: function Log(X : Float) : Float; external 'dmath';
9135: { Natural log }
9136: function Log2(X : Float) : Float; external 'dmath';
9137: { Log, base 2 }
9138: function Log10(X : Float) : Float; external 'dmath';
9139: { Decimal log }
9140: function LogA(X, A : Float) : Float; external 'dmath';
9141: { Log, base A }
9142: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9143: { X^N }
9144: function Power(X, Y : Float) : Float; external 'dmath';
9145: { X^Y, X >= 0 }
9146: { ------------------------------------------------------------
9147:   Trigonometric functions
9148:   ------------------------------------------------------------ }
9149: function Pythag(X, Y : Float) : Float; external 'dmath';
9150: { Sqrt(X^2 + Y^2) }
9151: function FixAngle(Theta : Float) : Float; external 'dmath';
9152: { Set Theta in -Pi..Pi }
9153: function Tan(X : Float) : Float; external 'dmath';
9154: { Tangent }
9155: function ArcSin(X : Float) : Float; external 'dmath';
9156: { Arc sinus }
9157: function ArcCos(X : Float) : Float; external 'dmath';
9158: { Arc cosinus }
9159: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9160: { Angle (Ox, OM) with M(X,Y) }
9161: { ------------------------------------------------------------
9162:   Hyperbolic functions
9163:   ------------------------------------------------------------ }
```

```
9164: function Sinh(X : Float) : Float; external 'dmath';
9165: { Hyperbolic sine }
9166: function Cosh(X : Float) : Float; external 'dmath';
9167: { Hyperbolic cosine }
9168: function Tanh(X : Float) : Float; external 'dmath';
9169: { Hyperbolic tangent }
9170: function ArcSinh(X : Float) : Float; external 'dmath';
9171: { Inverse hyperbolic sine }
9172: function ArcCosh(X : Float) : Float; external 'dmath';
9173: { Inverse hyperbolic cosine }
9174: function ArcTanh(X : Float) : Float; external 'dmath';
9175: { Inverse hyperbolic tangent }
9176: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9177: { Sinh & Cosh }
9178: { ---------------------------------------------------------------
9179:   Gamma function and related functions
9180:   --------------------------------------------------------------- }
9181: function Fact(N : Integer) : Float; external 'dmath';
9182: { Factorial }
9183: function SgnGamma(X : Float) : Integer; external 'dmath';
9184: { Sign of Gamma function }
9185: function Gamma(X : Float) : Float; external 'dmath';
9186: { Gamma function }
9187: function LnGamma(X : Float) : Float; external 'dmath';
9188: { Logarithm of Gamma function }
9189: function Stirling(X : Float) : Float; external 'dmath';
9190: { Stirling's formula for the Gamma function }
9191: function StirLog(X : Float) : Float; external 'dmath';
9192: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9193: function DiGamma(X : Float ) : Float; external 'dmath';
9194: { Digamma function }
9195: function TriGamma(X : Float ) : Float; external 'dmath';
9196: { Trigamma function }
9197: function IGamma(A, X : Float) : Float; external 'dmath';
9198: { Incomplete Gamma function}
9199: function JGamma(A, X : Float) : Float; external 'dmath';
9200: { Complement of incomplete Gamma function }
9201: function InvGamma(A, P : Float) : Float; external 'dmath';
9202: { Inverse of incomplete Gamma function }
9203: function Erf(X : Float) : Float; external 'dmath';
9204: { Error function }
9205: function Erfc(X : Float) : Float; external 'dmath';
9206: { Complement of error function }
9207: { ---------------------------------------------------------------
9208:   Beta function and related functions
9209:   --------------------------------------------------------------- }
9210: function Beta(X, Y : Float) : Float; external 'dmath';
9211: { Beta function }
9212: function IBeta(A, B, X : Float) : Float; external 'dmath';
9213: { Incomplete Beta function }
9214: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9215: { Inverse of incomplete Beta function }
9216: { ---------------------------------------------------------------
9217:   Lambert's function
9218:   --------------------------------------------------------------- }
9219: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9220:  ---------------------------------------------------------------
9221:   Binomial distribution
9222:   --------------------------------------------------------------- }
9223: function Binomial(N, K : Integer) : Float; external 'dmath';
9224: { Binomial coefficient C(N,K) }
9225: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9226: { Probability of binomial distribution }
9227: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9228: { Cumulative probability for binomial distrib. }
9229: { ---------------------------------------------------------------
9230:   Poisson distribution
9231:   --------------------------------------------------------------- }
9232: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9233: { Probability of Poisson distribution }
9234: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9235: { Cumulative probability for Poisson distrib. }
9236: { ---------------------------------------------------------------
9237:   Exponential distribution
9238:   --------------------------------------------------------------- }
9239: function DExpo(A, X : Float) : Float; external 'dmath';
9240: { Density of exponential distribution with parameter A }
9241: function FExpo(A, X : Float) : Float; external 'dmath';
9242: { Cumulative probability function for exponential dist. with parameter A }
9243: { ---------------------------------------------------------------
9244:   Standard normal distribution
9245:   --------------------------------------------------------------- }
9246: function DNorm(X : Float) : Float; external 'dmath';
9247: { Density of standard normal distribution }
9248: function FNorm(X : Float) : Float; external 'dmath';
9249: { Cumulative probability for standard normal distrib. }
9250: function PNorm(X : Float) : Float; external 'dmath';
9251: { Prob(|U| > X) for standard normal distrib. }
9252: function InvNorm(P : Float) : Float; external 'dmath';
```

```
9253: { Inverse of standard normal distribution }
9254: { ------------------------------------------------------------
9255:   Student's distribution
9256:   ------------------------------------------------------------ }
9257: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9258: { Density of Student distribution with Nu d.o.f. }
9259: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9260: { Cumulative probability for Student distrib. with Nu d.o.f. }
9261: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9262: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9263: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9264: { Inverse of Student's t-distribution function }
9265: { ------------------------------------------------------------
9266:   Khi-2 distribution
9267:   ------------------------------------------------------------ }
9268: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9269: { Density of Khi-2 distribution with Nu d.o.f. }
9270: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9271: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9272: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9273: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9274: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9275: { Inverse of Khi-2 distribution function }
9276: { ------------------------------------------------------------
9277:   Fisher-Snedecor distribution
9278:   ------------------------------------------------------------ }
9279: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9280: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9281: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9282: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9283: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9284: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9285: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9286: { Inverse of Snedecor's F-distribution function }
9287: { ------------------------------------------------------------
9288:   Beta distribution
9289:   ------------------------------------------------------------ }
9290: function DBeta(A, B, X : Float) : Float; external 'dmath';
9291: { Density of Beta distribution with parameters A and B }
9292: function FBeta(A, B, X : Float) : Float; external 'dmath';
9293: { Cumulative probability for Beta distrib. with param. A and B }
9294: { ------------------------------------------------------------
9295:   Gamma distribution
9296:   ------------------------------------------------------------ }
9297: function DGamma(A, B, X : Float) : Float; external 'dmath';
9298: { Density of Gamma distribution with parameters A and B }
9299: function FGamma(A, B, X : Float) : Float; external 'dmath';
9300: { Cumulative probability for Gamma distrib. with param. A and B }
9301: { ------------------------------------------------------------
9302:   Expression evaluation
9303:   ------------------------------------------------------------ }
9304: function InitEval : Integer; external 'dmath';
9305: { Initializes built-in functions and returns their number }
9306: function Eval(ExpressionString : String) : Float; external 'dmath';
9307: { Evaluates an expression at run-time }
9308: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9309: { Assigns a value to a variable }
9310: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9311: { Adds a function to the parser }
9312: { ------------------------------------------------------------
9313:   Matrices and linear equations
9314:   ------------------------------------------------------------ }
9315: procedure GaussJordan(A           : TMatrix;
9316:                       Lb, Ub1, Ub2 : Integer;
9317:                       var Det      : Float); external 'dmath';
9318: { Transforms a matrix according to the Gauss-Jordan method }
9319: procedure LinEq(A       : TMatrix;
9320:                 B       : TVector;
9321:                 Lb, Ub  : Integer;
9322:                 var Det : Float); external 'dmath';
9323: { Solves a linear system according to the Gauss-Jordan method }
9324: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9325: { Cholesky factorization of a positive definite symmetric matrix }
9326: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9327: { LU decomposition }
9328: procedure LU_Solve(A      : TMatrix;
9329:                    B      : TVector;
9330:                    Lb, Ub : Integer;
9331:                    X      : TVector); external 'dmath';
9332: { Solution of linear system from LU decomposition }
9333: procedure QR_Decomp(A           : TMatrix;
9334:                     Lb, Ub1, Ub2 : Integer;
9335:                     R           : TMatrix); external 'dmath';
9336: { QR decomposition }
9337: procedure QR_Solve(Q, R         : TMatrix;
9338:                    B           : TVector;
9339:                    Lb, Ub1, Ub2 : Integer;
9340:                    X           : TVector); external 'dmath';
9341: { Solution of linear system from QR decomposition }
```

```
9342: procedure SV_Decomp(A            : TMatrix;
9343:                     Lb, Ub1, Ub2 : Integer;
9344:                     S            : TVector;
9345:                     V            : TMatrix); external 'dmath';
9346: { Singular value decomposition }
9347: procedure SV_SetZero(S      : TVector;
9348:                      Lb, Ub : Integer;
9349:                      Tol    : Float); external 'dmath';
9350: { Set lowest singular values to zero }
9351: procedure SV_Solve(U            : TMatrix;
9352:                    S            : TVector;
9353:                    V            : TMatrix;
9354:                    B            : TVector;
9355:                    Lb, Ub1, Ub2 : Integer;
9356:                    X            : TVector); external 'dmath';
9357: { Solution of linear system from SVD }
9358: procedure SV_Approx(U            : TMatrix;
9359:                     S            : TVector;
9360:                     V            : TMatrix;
9361:                     Lb, Ub1, Ub2 : Integer;
9362:                     A            : TMatrix); external 'dmath';
9363: { Matrix approximation from SVD }
9364: procedure EigenVals(A      : TMatrix;
9365:                     Lb, Ub : Integer;
9366:                     Lambda : TCompVector); external 'dmath';
9367: { Eigenvalues of a general square matrix }
9368: procedure EigenVect(A      : TMatrix;
9369:                     Lb, Ub : Integer;
9370:                     Lambda : TCompVector;
9371:                     V      : TMatrix); external 'dmath';
9372: { Eigenvalues and eigenvectors of a general square matrix }
9373: procedure EigenSym(A      : TMatrix;
9374:                    Lb, Ub : Integer;
9375:                    Lambda : TVector;
9376:                    V      : TMatrix); external 'dmath';
9377: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9378: procedure Jacobi(A               : TMatrix;
9379:                  Lb, Ub, MaxIter : Integer;
9380:                  Tol             : Float;
9381:                  Lambda          : TVector;
9382:                  V               : TMatrix); external 'dmath';
9383: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9384: { ---------------------------------------------------------------
9385:   Optimization
9386:   --------------------------------------------------------------- }
9387: procedure MinBrack(Func                    : TFunc;
9388:                    var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9389: { Brackets a minimum of a function }
9390: procedure GoldSearch(Func           : TFunc;
9391:                      A, B           : Float;
9392:                      MaxIter        : Integer;
9393:                      Tol            : Float;
9394:                      var Xmin, Ymin : Float); external 'dmath';
9395: { Minimization of a function of one variable (golden search) }
9396: procedure LinMin(Func      : TFuncNVar;
9397:                  X, DeltaX : TVector;
9398:                  Lb, Ub    : Integer;
9399:                  var R     : Float;
9400:                  MaxIter   : Integer;
9401:                  Tol       : Float;
9402:                  var F_min : Float); external 'dmath';
9403: { Minimization of a function of several variables along a line }
9404: procedure Newton(Func      : TFuncNVar;
9405:                  HessGrad  : THessGrad;
9406:                  X         : TVector;
9407:                  Lb, Ub    : Integer;
9408:                  MaxIter   : Integer;
9409:                  Tol       : Float;
9410:                  var F_min : Float;
9411:                  G         : TVector;
9412:                  H_inv     : TMatrix;
9413:                  var Det   : Float); external 'dmath';
9414: { Minimization of a function of several variables (Newton's method) }
9415: procedure SaveNewton(FileName : string); external 'dmath';
9416: { Save Newton iterations in a file }
9417: procedure Marquardt(Func      : TFuncNVar;
9418:                     HessGrad  : THessGrad;
9419:                     X         : TVector;
9420:                     Lb, Ub    : Integer;
9421:                     MaxIter   : Integer;
9422:                     Tol       : Float;
9423:                     var F_min : Float;
9424:                     G         : TVector;
9425:                     H_inv     : TMatrix;
9426:                     var Det   : Float); external 'dmath';
9427: { Minimization of a function of several variables (Marquardt's method) }
9428: procedure SaveMarquardt(FileName : string); external 'dmath';
9429: { Save Marquardt iterations in a file }
9430: procedure BFGS(Func      : TFuncNVar;
```

```
9431:               Gradient  : TGradient;
9432:               X         : TVector;
9433:               Lb, Ub    : Integer;
9434:               MaxIter   : Integer;
9435:               Tol       : Float;
9436:               var F_min : Float;
9437:               G         : TVector;
9438:               H_inv     : TMatrix); external 'dmath';
9439: { Minimization of a function of several variables (BFGS method) }
9440: procedure SaveBFGS(FileName : string); external 'dmath';
9441: { Save BFGS iterations in a file }
9442: procedure Simplex(Func      : TFuncNVar;
9443:                   X         : TVector;
9444:                   Lb, Ub    : Integer;
9445:                   MaxIter   : Integer;
9446:                   Tol       : Float;
9447:                   var F_min : Float); external 'dmath';
9448: { Minimization of a function of several variables (Simplex) }
9449: procedure SaveSimplex(FileName : string); external 'dmath';
9450: { Save Simplex iterations in a file }
9451: { ------------------------------------------------------------------
9452:   Nonlinear equations
9453:   ------------------------------------------------------------------ }
9454: procedure RootBrack(Func            : TFunc;
9455:                     var X, Y, FX, FY : Float); external 'dmath';
9456: { Brackets a root of function Func between X and Y }
9457: procedure Bisect(Func     : TFunc;
9458:                  var X, Y : Float;
9459:                  MaxIter  : Integer;
9460:                  Tol      : Float;
9461:                  var F    : Float); external 'dmath';
9462: { Bisection method }
9463: procedure Secant(Func     : TFunc;
9464:                  var X, Y : Float;
9465:                  MaxIter  : Integer;
9466:                  Tol      : Float;
9467:                  var F    : Float); external 'dmath';
9468: { Secant method }
9469: procedure NewtEq(Func, Deriv : TFunc;
9470:                  var X       : Float;
9471:                  MaxIter     : Integer;
9472:                  Tol         : Float;
9473:                  var F       : Float); external 'dmath';
9474: { Newton-Raphson method for a single nonlinear equation }
9475: procedure NewtEqs(Equations : TEquations;
9476:                   Jacobian  : TJacobian;
9477:                   X, F      : TVector;
9478:                   Lb, Ub    : Integer;
9479:                   MaxIter   : Integer;
9480:                   Tol       : Float); external 'dmath';
9481: { Newton-Raphson method for a system of nonlinear equations }
9482: procedure Broyden(Equations : TEquations;
9483:                   X, F      : TVector;
9484:                   Lb, Ub    : Integer;
9485:                   MaxIter   : Integer;
9486:                   Tol       : Float); external 'dmath';
9487: { Broyden's method for a system of nonlinear equations }
9488: { ------------------------------------------------------------------
9489:   Polynomials and rational fractions
9490:   ------------------------------------------------------------------ }
9491: function Poly(X    : Float;
9492:               Coef : TVector;
9493:               Deg  : Integer) : Float; external 'dmath';
9494: { Evaluates a polynomial }
9495: function RFrac(X          : Float;
9496:                Coef       : TVector;
9497:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9498: { Evaluates a rational fraction }
9499: function RootPol1(A, B  : Float;
9500:                   var X : Float) : Integer; external 'dmath';
9501: { Solves the linear equation A + B * X = 0 }
9502: function RootPol2(Coef : TVector;
9503:                   Z    : TCompVector) : Integer; external 'dmath';
9504: { Solves a quadratic equation }
9505: function RootPol3(Coef : TVector;
9506:                   Z    : TCompVector) : Integer; external 'dmath';
9507: { Solves a cubic equation }
9508: function RootPol4(Coef : TVector;
9509:                   Z    : TCompVector) : Integer; external 'dmath';
9510: { Solves a quartic equation }
9511: function RootPol(Coef : TVector;
9512:                  Deg  : Integer;
9513:                  Z    : TCompVector) : Integer; external 'dmath';
9514: { Solves a polynomial equation }
9515: function SetRealRoots(Deg : Integer;
9516:                       Z   : TCompVector;
9517:                       Tol : Float) : Integer; external 'dmath';
9518: { Set the imaginary part of a root to zero }
9519: procedure SortRoots(Deg : Integer;
```

```
                    Z   : TCompVector); external 'dmath';
{ Sorts the roots of a polynomial }
{ ------------------------------------------------------------------
  Numerical integration and differential equations
  ------------------------------------------------------------------ }
function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
{ Integration by trapezoidal rule }
function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
{ Integral from A to B }
function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
{ Integral from 0 to B }
function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
{ Convolution product at time T }
procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
{ Convolution by trapezoidal rule }
procedure RKF45(F                   : TDiffEqs;
                Neqn                : Integer;
                Y, Yp               : TVector;
                var T               : Float;
                Tout, RelErr, AbsErr : Float;
                var Flag            : Integer); external 'dmath';
{ Integration of a system of differential equations }
{ ------------------------------------------------------------------
  Fast Fourier Transform
  ------------------------------------------------------------------ }
procedure FFT(NumSamples        : Integer;
              InArray, OutArray : TCompVector); external 'dmath';
{ Fast Fourier Transform }
procedure IFFT(NumSamples        : Integer;
               InArray, OutArray : TCompVector); external 'dmath';
{ Inverse Fast Fourier Transform }
procedure FFT_Integer(NumSamples     : Integer;
                      RealIn, ImagIn : TIntVector;
                      OutArray       : TCompVector); external 'dmath';
{ Fast Fourier Transform for integer data }
procedure FFT_Integer_Cleanup; external 'dmath';
{ Clear memory after a call to FFT_Integer }
procedure CalcFrequency(NumSamples,
                        FrequencyIndex : Integer;
                        InArray        : TCompVector;
                        var FFT        : Complex); external 'dmath';
{ Direct computation of Fourier transform }
{ ------------------------------------------------------------------
  Random numbers
  ------------------------------------------------------------------ }
procedure SetRNG(RNG : RNG_Type); external 'dmath';
{ Select generator }
procedure InitGen(Seed : RNG_IntType); external 'dmath';
{ Initialize generator }
function IRanGen : RNG_IntType; external 'dmath';
{ 32-bit random integer in [-2^31 .. 2^31 - 1] }
function IRanGen31 : RNG_IntType; external 'dmath';
{ 31-bit random integer in [0 .. 2^31 - 1] }
function RanGen1 : Float; external 'dmath';
{ 32-bit random real in [0,1] }
function RanGen2 : Float; external 'dmath';
{ 32-bit random real in [0,1) }
function RanGen3 : Float; external 'dmath';
{ 32-bit random real in (0,1) }
function RanGen53 : Float; external 'dmath';
{ 53-bit random real in [0,1) }
procedure InitMWC(Seed : RNG_IntType); external 'dmath';
{ Initializes the 'Multiply with carry' random number generator }
function IRanMWC : RNG_IntType; external 'dmath';
{ Returns a 32 bit random number in [-2^31 ; 2^31-1] }
procedure InitMT(Seed : RNG_IntType); external 'dmath';
{ Initializes Mersenne Twister generator with a seed }
procedure InitMTbyArray(InitKey   : array of RNG_LongType;
                        KeyLength : Word); external 'dmath';
{ Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
function IRanMT : RNG_IntType; external 'dmath';
{ Random integer from MT generator }
procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
{ Initializes the UVAG generator with a string }
procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
{ Initializes the UVAG generator with an integer }
function IRanUVAG : RNG_IntType; external 'dmath';
{ Random integer from UVAG generator }
function RanGaussStd : Float; external 'dmath';
{ Random number from standard normal distribution }
function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
{ Random number from normal distrib. with mean Mu and S. D. Sigma }
procedure RanMult(M      : TVector; L      : TMatrix;
                  Lb, Ub : Integer;
                  X      : TVector); external 'dmath';
{ Random vector from multinormal distribution (correlated) }
procedure RanMultIndep(M, S   : TVector;
                       Lb, Ub : Integer;
                       X      : TVector); external 'dmath';
```

```
9609: { Random vector from multinormal distribution (uncorrelated) }
9610: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9611: { Initializes Metropolis-Hastings parameters }
9612: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9613: { Returns Metropolis-Hastings parameters }
9614: procedure Hastings(Func      : TFuncNVar;
9615:                    T         : Float;
9616:                    X         : TVector;
9617:                    V         : TMatrix;
9618:                    Lb, Ub    : Integer;
9619:                    Xmat      : TMatrix;
9620:                    X_min     : TVector;
9621:                    var F_min : Float); external 'dmath';
9622: { Simulation of a probability density function by Metropolis-Hastings }
9623: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9624: { Initializes Simulated Annealing parameters }
9625: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9626: { Initializes log file }
9627: procedure SimAnn(Func          : TFuncNVar;
9628:                  X, Xmin, Xmax : TVector;
9629:                  Lb, Ub        : Integer;
9630:                  var F_min     : Float); external 'dmath';
9631: { Minimization of a function of several var. by simulated annealing }
9632: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9633: { Initializes Genetic Algorithm parameters }
9634: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9635: { Initializes log file }
9636: procedure GenAlg(Func          : TFuncNVar;
9637:                  X, Xmin, Xmax : TVector;
9638:                  Lb, Ub        : Integer;
9639:                  var F_min     : Float); external 'dmath';
9640: { Minimization of a function of several var. by genetic algorithm }
9641: { ---------------------------------------------------------------
9642:   Statistics
9643:   --------------------------------------------------------------- }
9644: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9645: { Mean of sample X }
9646: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9647: { Minimum of sample X }
9648: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9649: { Maximum of sample X }
9650: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9651: { Median of sample X }
9652: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9653: { Standard deviation estimated from sample X }
9654: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9655: { Standard deviation of population }
9656: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9657: { Correlation coefficient }
9658: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9659: { Skewness of sample X }
9660: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9661: { Kurtosis of sample X }
9662: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9663: { Quick sort (ascending order) }
9664: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9665: { Quick sort (descending order) }
9666: procedure Interval(X1, X2             : Float;
9667:                    MinDiv, MaxDiv     : Integer;
9668:                    var Min, Max, Step : Float); external 'dmath';
9669: { Determines an interval for a set of values }
9670: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9671:                     var XMin, XMax, XStep : Float); external 'dmath';
9672: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9673: procedure StudIndep(N1, N2         : Integer;
9674:                     M1, M2, S1, S2 : Float;
9675:                     var T          : Float;
9676:                     var DoF        : Integer); external 'dmath';
9677: { Student t-test for independent samples }
9678: procedure StudPaired(X, Y    : TVector;
9679:                      Lb, Ub  : Integer;
9680:                      var T   : Float;
9681:                      var DoF : Integer); external 'dmath';
9682: { Student t-test for paired samples }
9683: procedure AnOVa1(Ns               : Integer;
9684:                  N                : TIntVector;
9685:                  M, S             : TVector;
9686:                  var V_f, V_r, F  : Float;
9687:                  var DoF_f, DoF_r : Integer); external 'dmath';
9688: { One-way analysis of variance }
9689: procedure AnOVa2(NA, NB, Nobs : Integer;
9690:                  M, S         : TMatrix;
9691:                  V, F         : TVector;
9692:                  DoF          : TIntVector); external 'dmath';
9693: { Two-way analysis of variance }
9694: procedure Snedecor(N1, N2         : Integer;
9695:                    S1, S2         : Float;
9696:                    var F          : Float;
9697:                    var DoF1, DoF2 : Integer); external 'dmath';
```

```
9698: { Snedecor's F-test (comparison of two variances) }
9699: procedure Bartlett(Ns       : Integer;
9700:                    N        : TIntVector;
9701:                    S        : TVector;
9702:                    var Khi2 : Float;
9703:                    var DoF  : Integer); external 'dmath';
9704: { Bartlett's test (comparison of several variances) }
9705: procedure Mann_Whitney(N1, N2     : Integer;
9706:                        X1, X2     : TVector;
9707:                        var U, Eps : Float); external 'dmath';
9708: { Mann-Whitney test}
9709: procedure Wilcoxon(X, Y       : TVector;
9710:                    Lb, Ub     : Integer;
9711:                    var Ndiff  : Integer;
9712:                    var T, Eps : Float); external 'dmath';
9713: { Wilcoxon test }
9714: procedure Kruskal_Wallis(Ns      : Integer;
9715:                          N       : TIntVector;
9716:                          X       : TMatrix;
9717:                          var H   : Float;
9718:                          var DoF : Integer); external 'dmath';
9719: { Kruskal-Wallis test }
9720: procedure Khi2_Conform(N_cls    : Integer;
9721:                        N_estim  : Integer;
9722:                        Obs      : TIntVector;
9723:                        Calc     : TVector;
9724:                        var Khi2 : Float;
9725:                        var DoF  : Integer); external 'dmath';
9726: { Khi-2 test for conformity }
9727: procedure Khi2_Indep(N_lin    : Integer;
9728:                      N_col    : Integer;
9729:                      Obs      : TIntMatrix;
9730:                      var Khi2 : Float;
9731:                      var DoF  : Integer); external 'dmath';
9732: { Khi-2 test for independence }
9733: procedure Woolf_Conform(N_cls   : Integer;
9734:                         N_estim : Integer;
9735:                         Obs     : TIntVector;
9736:                         Calc    : TVector;
9737:                         var G   : Float;
9738:                         var DoF : Integer); external 'dmath';
9739: { Woolf's test for conformity }
9740: procedure Woolf_Indep(N_lin   : Integer;
9741:                       N_col   : Integer;
9742:                       Obs     : TIntMatrix;
9743:                       var G   : Float;
9744:                       var DoF : Integer); external 'dmath';
9745: { Woolf's test for independence }
9746: procedure DimStatClassVector(var C : TStatClassVector;
9747:                              Ub    : Integer); external 'dmath';
9748: { Allocates an array of statistical classes: C[0..Ub] }
9749: procedure Distrib(X       : TVector;
9750:                   Lb, Ub  : Integer;
9751:                   A, B, H : Float;
9752:                   C       : TStatClassVector); external 'dmath';
9753: { Distributes an array X[Lb..Ub] into statistical classes }
9754: { ----------------------------------------------------------------
9755:   Linear / polynomial regression
9756:   ---------------------------------------------------------------- }
9757: procedure LinFit(X, Y   : TVector;
9758:                  Lb, Ub : Integer;
9759:                  B      : TVector;
9760:                  V      : TMatrix); external 'dmath';
9761: { Linear regression : Y = B(0) + B(1) * X }
9762: procedure WLinFit(X, Y, S : TVector;
9763:                   Lb, Ub  : Integer;
9764:                   B       : TVector;
9765:                   V       : TMatrix); external 'dmath';
9766: { Weighted linear regression : Y = B(0) + B(1) * X }
9767: procedure SVDLinFit(X, Y   : TVector;
9768:                     Lb, Ub : Integer;
9769:                     SVDTol : Float;
9770:                     B      : TVector;
9771:                     V      : TMatrix); external 'dmath';
9772: { Unweighted linear regression by singular value decomposition }
9773: procedure WSVDLinFit(X, Y, S : TVector;
9774:                      Lb, Ub  : Integer;
9775:                      SVDTol  : Float;
9776:                      B       : TVector;
9777:                      V       : TMatrix); external 'dmath';
9778: { Weighted linear regression by singular value decomposition }
9779: procedure MulFit(X            : TMatrix;
9780:                  Y            : TVector;
9781:                  Lb, Ub, Nvar : Integer;
9782:                  ConsTerm     : Boolean;
9783:                  B            : TVector;
9784:                  V            : TMatrix); external 'dmath';
9785: { Multiple linear regression by Gauss-Jordan method }
9786: procedure WMulFit(X            : TMatrix;
```

```
9787:                   Y, S         : TVector;
9788:                   Lb, Ub, Nvar : Integer;
9789:                   ConsTerm     : Boolean;
9790:                   B            : TVector;
9791:                   V            : TMatrix); external 'dmath';
9792: { Weighted multiple linear regression by Gauss-Jordan method }
9793: procedure SVDFit(X            : TMatrix;
9794:                  Y            : TVector;
9795:                  Lb, Ub, Nvar : Integer;
9796:                  ConsTerm     : Boolean;
9797:                  SVDTol       : Float;
9798:                  B            : TVector;
9799:                  V            : TMatrix); external 'dmath';
9800: { Multiple linear regression by singular value decomposition }
9801: procedure WSVDFit(X            : TMatrix;
9802:                   Y, S         : TVector;
9803:                   Lb, Ub, Nvar : Integer;
9804:                   ConsTerm     : Boolean;
9805:                   SVDTol       : Float;
9806:                   B            : TVector;
9807:                   V            : TMatrix); external 'dmath';
9808: { Weighted multiple linear regression by singular value decomposition }
9809: procedure PolFit(X, Y        : TVector;
9810:                  Lb, Ub, Deg : Integer;
9811:                  B           : TVector;
9812:                  V           : TMatrix); external 'dmath';
9813: { Polynomial regression by Gauss-Jordan method }
9814: procedure WPolFit(X, Y, S     : TVector;
9815:                   Lb, Ub, Deg : Integer;
9816:                   B           : TVector;
9817:                   V           : TMatrix); external 'dmath';
9818: { Weighted polynomial regression by Gauss-Jordan method }
9819: procedure SVDPolFit(X, Y        : TVector;
9820:                     Lb, Ub, Deg : Integer;
9821:                     SVDTol      : Float;
9822:                     B           : TVector;
9823:                     V           : TMatrix); external 'dmath';
9824: { Unweighted polynomial regression by singular value decomposition }
9825: procedure WSVDPolFit(X, Y, S     : TVector;
9826:                      Lb, Ub, Deg : Integer;
9827:                      SVDTol      : Float;
9828:                      B           : TVector;
9829:                      V           : TMatrix); external 'dmath';
9830: { Weighted polynomial regression by singular value decomposition }
9831: procedure RegTest(Y, Ycalc : TVector;
9832:                   LbY, UbY : Integer;
9833:                   V        : TMatrix;
9834:                   LbV, UbV : Integer;
9835:                   var Test : TRegTest); external 'dmath';
9836: { Test of unweighted regression }
9837: procedure WRegTest(Y, Ycalc, S : TVector;
9838:                    LbY, UbY    : Integer;
9839:                    V           : TMatrix;
9840:                    LbV, UbV    : Integer;
9841:                    var Test    : TRegTest); external 'dmath';
9842: { Test of weighted regression }
9843: { ------------------------------------------------------------------
9844:   Nonlinear regression
9845:   ------------------------------------------------------------------ }
9846: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9847: { Sets the optimization algorithm for nonlinear regression }
9848: function GetOptAlgo : TOptAlgo; external 'dmath';
9849: { Returns the optimization algorithm }
9850: procedure SetMaxParam(N : Byte); external 'dmath';
9851: { Sets the maximum number of regression parameters for nonlinear regression }
9852: function GetMaxParam : Byte; external 'dmath';
9853: { Returns the maximum number of regression parameters for nonlinear regression }
9854: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9855: { Sets the bounds on the I-th regression parameter }
9856: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9857: { Returns the bounds on the I-th regression parameter }
9858: procedure NLFit(RegFunc   : TRegFunc;
9859:                 DerivProc : TDerivProc;
9860:                 X, Y      : TVector;
9861:                 Lb, Ub    : Integer;
9862:                 MaxIter   : Integer;
9863:                 Tol       : Float;
9864:                 B         : TVector;
9865:                 FirstPar,
9866:                 LastPar   : Integer;
9867:                 V         : TMatrix); external 'dmath';
9868: { Unweighted nonlinear regression }
9869: procedure WNLFit(RegFunc   : TRegFunc;
9870:                  DerivProc : TDerivProc;
9871:                  X, Y, S   : TVector;
9872:                  Lb, Ub    : Integer;
9873:                  MaxIter   : Integer;
9874:                  Tol       : Float;
9875:                  B         : TVector;
```

```
9876:                 FirstPar,
9877:                 LastPar   : Integer;
9878:                 V         : TMatrix); external 'dmath';
9879: { Weighted nonlinear regression }
9880: procedure SetMCFile(FileName : String); external 'dmath';
9881: { Set file for saving MCMC simulations }
9882: procedure SimFit(RegFunc   : TRegFunc;
9883:                 X, Y      : TVector;
9884:                 Lb, Ub    : Integer;
9885:                 B         : TVector;
9886:                 FirstPar,
9887:                 LastPar   : Integer;
9888:                 V         : TMatrix); external 'dmath';
9889: { Simulation of unweighted nonlinear regression by MCMC }
9890: procedure WSimFit(RegFunc   : TRegFunc;
9891:                  X, Y, S   : TVector;
9892:                  Lb, Ub    : Integer;
9893:                  B         : TVector;
9894:                  FirstPar,
9895:                  LastPar   : Integer;
9896:                  V         : TMatrix); external 'dmath';
9897: { Simulation of weighted nonlinear regression by MCMC }
9898: { ------------------------------------------------------------------
9899:   Nonlinear regression models
9900:   ------------------------------------------------------------------ }
9901: procedure FracFit(X, Y       : TVector;
9902:                  Lb, Ub     : Integer;
9903:                  Deg1, Deg2 : Integer;
9904:                  ConsTerm   : Boolean;
9905:                  MaxIter    : Integer;
9906:                  Tol        : Float;
9907:                  B          : TVector;
9908:                  V          : TMatrix); external 'dmath';
9909: { Unweighted fit of rational fraction }
9910: procedure WFracFit(X, Y, S    : TVector;
9911:                   Lb, Ub     : Integer;
9912:                   Deg1, Deg2 : Integer;
9913:                   ConsTerm   : Boolean;
9914:                   MaxIter    : Integer;
9915:                   Tol        : Float;
9916:                   B          : TVector;
9917:                   V          : TMatrix); external 'dmath';
9918: { Weighted fit of rational fraction }
9919:
9920: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9921: { Returns the value of the rational fraction at point X }
9922: procedure ExpFit(X, Y         : TVector;
9923:                 Lb, Ub, Nexp : Integer;
9924:                 ConsTerm     : Boolean;
9925:                 MaxIter      : Integer;
9926:                 Tol          : Float;
9927:                 B            : TVector;
9928:                 V            : TMatrix); external 'dmath';
9929: { Unweighted fit of sum of exponentials }
9930: procedure WExpFit(X, Y, S      : TVector;
9931:                  Lb, Ub, Nexp : Integer;
9932:                  ConsTerm     : Boolean;
9933:                  MaxIter      : Integer;
9934:                  Tol          : Float;
9935:                  B            : TVector;
9936:                  V            : TMatrix); external 'dmath';
9937: { Weighted fit of sum of exponentials }
9938: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9939: { Returns the value of the regression function at point X }
9940: procedure IncExpFit(X, Y     : TVector;
9941:                    Lb, Ub    : Integer;
9942:                    ConsTerm : Boolean;
9943:                    MaxIter  : Integer;
9944:                    Tol      : Float;
9945:                    B        : TVector;
9946:                    V        : TMatrix); external 'dmath';
9947: { Unweighted fit of model of increasing exponential }
9948: procedure WIncExpFit(X, Y, S  : TVector;
9949:                     Lb, Ub   : Integer;
9950:                     ConsTerm : Boolean;
9951:                     MaxIter  : Integer;
9952:                     Tol      : Float;
9953:                     B        : TVector;
9954:                     V        : TMatrix); external 'dmath';
9955: { Weighted fit of increasing exponential }
9956: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9957: { Returns the value of the regression function at point X }
9958: procedure ExpLinFit(X, Y    : TVector;
9959:                    Lb, Ub  : Integer;
9960:                    MaxIter : Integer;
9961:                    Tol     : Float;
9962:                    B       : TVector;
9963:                    V       : TMatrix); external 'dmath';
9964: { Unweighted fit of the "exponential + linear" model }
```

```
 9965: procedure WExpLinFit(X, Y, S : TVector;
 9966:                      Lb, Ub  : Integer;
 9967:                      MaxIter : Integer;
 9968:                      Tol     : Float;
 9969:                      B       : TVector;
 9970:                      V       : TMatrix); external 'dmath';
 9971: { Weighted fit of the "exponential + linear" model }
 9972:
 9973: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9974: { Returns the value of the regression function at point X }
 9975: procedure MichFit(X, Y    : TVector;
 9976:                   Lb, Ub  : Integer;
 9977:                   MaxIter : Integer;
 9978:                   Tol     : Float;
 9979:                   B       : TVector;
 9980:                   V       : TMatrix); external 'dmath';
 9981: { Unweighted fit of Michaelis equation }
 9982: procedure WMichFit(X, Y, S : TVector;
 9983:                    Lb, Ub  : Integer;
 9984:                    MaxIter : Integer;
 9985:                    Tol     : Float;
 9986:                    B       : TVector;
 9987:                    V       : TMatrix); external 'dmath';
 9988: { Weighted fit of Michaelis equation }
 9989: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9990: { Returns the value of the Michaelis equation at point X }
 9991: procedure MintFit(X, Y    : TVector;
 9992:                   Lb, Ub  : Integer;
 9993:                   MintVar : TMintVar;
 9994:                   Fit_S0  : Boolean;
 9995:                   MaxIter : Integer;
 9996:                   Tol     : Float;
 9997:                   B       : TVector;
 9998:                   V       : TMatrix); external 'dmath';
 9999: { Unweighted fit of the integrated Michaelis equation }
10000: procedure WMintFit(X, Y, S : TVector;
10001:                    Lb, Ub  : Integer;
10002:                    MintVar : TMintVar;
10003:                    Fit_S0  : Boolean;
10004:                    MaxIter : Integer;
10005:                    Tol     : Float;
10006:                    B       : TVector;
10007:                    V       : TMatrix); external 'dmath';
10008: { Weighted fit of the integrated Michaelis equation }
10009: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10010: { Returns the value of the integrated Michaelis equation at point X }
10011: procedure HillFit(X, Y    : TVector;
10012:                   Lb, Ub  : Integer;
10013:                   MaxIter : Integer;
10014:                   Tol     : Float;
10015:                   B       : TVector;
10016:                   V       : TMatrix); external 'dmath';
10017: { Unweighted fit of Hill equation }
10018: procedure WHillFit(X, Y, S : TVector;
10019:                    Lb, Ub  : Integer;
10020:                    MaxIter : Integer;
10021:                    Tol     : Float;
10022:                    B       : TVector;
10023:                    V       : TMatrix); external 'dmath';
10024: { Weighted fit of Hill equation }
10025: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10026: { Returns the value of the Hill equation at point X }
10027: procedure LogiFit(X, Y     : TVector;
10028:                   Lb, Ub   : Integer;
10029:                   ConsTerm : Boolean;
10030:                   General  : Boolean;
10031:                   MaxIter  : Integer;
10032:                   Tol      : Float;
10033:                   B        : TVector;
10034:                   V        : TMatrix); external 'dmath';
10035: { Unweighted fit of logistic function }
10036: procedure WLogiFit(X, Y, S  : TVector;
10037:                    Lb, Ub   : Integer;
10038:                    ConsTerm : Boolean;
10039:                    General  : Boolean;
10040:                    MaxIter  : Integer;
10041:                    Tol      : Float;
10042:                    B        : TVector;
10043:                    V        : TMatrix); external 'dmath';
10044: { Weighted fit of logistic function }
10045: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10046: { Returns the value of the logistic function at point X }
10047: procedure PKFit(X, Y    : TVector;
10048:                 Lb, Ub  : Integer;
10049:                 MaxIter : Integer;
10050:                 Tol     : Float;
10051:                 B       : TVector;
10052:                 V       : TMatrix); external 'dmath';
10053: { Unweighted fit of the acid-base titration curve }
```

```
10054: procedure WPKFit(X, Y, S : TVector;
10055:                  Lb, Ub  : Integer;
10056:                  MaxIter : Integer;
10057:                  Tol     : Float;
10058:                  B       : TVector;
10059:                  V       : TMatrix); external 'dmath';
10060: { Weighted fit of the acid-base titration curve }
10061: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10062: { Returns the value of the acid-base titration function at point X }
10063: procedure PowFit(X, Y    : TVector;
10064:                  Lb, Ub  : Integer;
10065:                  MaxIter : Integer;
10066:                  Tol     : Float;
10067:                  B       : TVector;
10068:                  V       : TMatrix); external 'dmath';
10069: { Unweighted fit of power function }
10070: procedure WPowFit(X, Y, S : TVector;
10071:                   Lb, Ub  : Integer;
10072:                   MaxIter : Integer;
10073:                   Tol     : Float;
10074:                   B       : TVector;
10075:                   V       : TMatrix); external 'dmath';
10076: { Weighted fit of power function }
10077:
10078: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10079: { Returns the value of the power function at point X }
10080: procedure GammaFit(X, Y    : TVector;
10081:                    Lb, Ub  : Integer;
10082:                    MaxIter : Integer;
10083:                    Tol     : Float;
10084:                    B       : TVector;
10085:                    V       : TMatrix); external 'dmath';
10086: { Unweighted fit of gamma distribution function }
10087: procedure WGammaFit(X, Y, S : TVector;
10088:                     Lb, Ub  : Integer;
10089:                     MaxIter : Integer;
10090:                     Tol     : Float;
10091:                     B       : TVector;
10092:                     V       : TMatrix); external 'dmath';
10093: { Weighted fit of gamma distribution function }
10094: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10095: { Returns the value of the gamma distribution function at point X }
10096: { ------------------------------------------------------------------
10097:   Principal component analysis
10098:   ------------------------------------------------------------------ }
10099: procedure VecMean(X            : TMatrix;
10100:                   Lb, Ub, Nvar : Integer;
10101:                   M            : TVector); external 'dmath';
10102: { Computes the mean vector M from matrix X }
10103: procedure VecSD(X            : TMatrix;
10104:                 Lb, Ub, Nvar : Integer;
10105:                 M, S         : TVector); external 'dmath';
10106: { Computes the vector of standard deviations S from matrix X }
10107: procedure MatVarCov(X            : TMatrix;
10108:                     Lb, Ub, Nvar : Integer;
10109:                     M            : TVector;
10110:                     V            : TMatrix); external 'dmath';
10111: { Computes the variance-covariance matrix V from matrix X }
10112: procedure MatCorrel(V    : TMatrix;
10113:                     Nvar : Integer;
10114:                     R    : TMatrix); external 'dmath';
10115: { Computes the correlation matrix R from the var-cov matrix V }
10116: procedure PCA(R      : TMatrix;
10117:               Nvar   : Integer;
10118:               Lambda : TVector;
10119:               C, Rc  : TMatrix); external 'dmath';
10120: { Performs a principal component analysis of the correlation matrix R }
10121: procedure ScaleVar(X            : TMatrix;
10122:                    Lb, Ub, Nvar : Integer;
10123:                    M, S         : TVector;
10124:                    Z            : TMatrix); external 'dmath';
10125: { Scales a set of variables by subtracting means and dividing by SD's }
10126: procedure PrinFac(Z            : TMatrix;
10127:                   Lb, Ub, Nvar : Integer;
10128:                   C, F         : TMatrix); external 'dmath';
10129: { Computes principal factors }
10130: { ------------------------------------------------------------------
10131:   Strings
10132:   ------------------------------------------------------------------ }
10133: function LTrim(S : String) : String; external 'dmath';
10134: { Removes leading blanks }
10135: function RTrim(S : String) : String; external 'dmath';
10136: { Removes trailing blanks }
10137: function Trim(S : String) : String; external 'dmath';
10138: { Removes leading and trailing blanks }
10139: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10140: { Returns a string made of character C repeated N times }
10141: function RFill(S : String; L : Byte) : String; external 'dmath';
10142: { Completes string S with trailing blanks for a total length L }
```

```
10143: function LFill(S : String; L : Byte) : String; external 'dmath';
10144: { Completes string S with leading blanks for a total length L }
10145: function CFill(S : String; L : Byte) : String; external 'dmath';
10146: { Centers string S on a total length L }
10147: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10148: { Replaces in string S all the occurences of C1 by C2 }
10149: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10150: { Extracts a field from a string }
10151: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10152: { Parses a string into its constitutive fields }
10153: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10154: { Sets the numeric format }
10155: function FloatStr(X : Float) : String; external 'dmath';
10156: { Converts a real to a string according to the numeric format }
10157: function IntStr(N : LongInt) : String; external 'dmath';
10158: { Converts an integer to a string }
10159: function CompStr(Z : Complex) : String; external 'dmath';
10160: { Converts a complex number to a string }
10161: {$IFDEF DELPHI}
10162: function StrDec(S : String) : String; external 'dmath';
10163: { Set decimal separator to the symbol defined in SysUtils }
10164: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10165: { Test if a string represents a number and returns it in X }
10166: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10167: { Reads a floating point number from an Edit control }
10168: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10169: { Writes a floating point number in a text file }
10170: {$ENDIF}
10171: { ------------------------------------------------------------------
10172:   BGI / Delphi graphics
10173:   ------------------------------------------------------------------ }
10174: function InitGraphics
10175: {$IFDEF DELPHI}
10176: (Width, Height : Integer) : Boolean;
10177: {$ELSE}
10178: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10179: { Enters graphic mode }
10180: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10181:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10182: { Sets the graphic window }
10183: procedure SetOxScale(Scale               : TScale;
10184:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10185: { Sets the scale on the Ox axis }
10186: procedure SetOyScale(Scale               : TScale;
10187:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10188: { Sets the scale on the Oy axis }
10189: procedure GetOxScale(var Scale               : TScale;
10190:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10191: { Returns the scale on the Ox axis }
10192: procedure GetOyScale(var Scale               : TScale;
10193:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10194: { Returns the scale on the Oy axis }
10195: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10196: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10197: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10198: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10199: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10200: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10201: {$IFNDEF DELPHI}
10202: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10203: { Sets the font for the main graph title }
10204: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10205: { Sets the font for the Ox axis (title and labels) }
10206: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10207: { Sets the font for the Oy axis (title and labels) }
10208: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10209: { Sets the font for the legends }
10210: procedure SetClipping(Clip : Boolean); external 'dmath';
10211: { Determines whether drawings are clipped at the current viewport
10212:   boundaries, according to the value of the Boolean parameter Clip }
10213: {$ENDIF}
10214: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10215: { Plots the horizontal axis }
10216: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10217: { Plots the vertical axis }
10218: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10219: { Plots a grid on the graph }
10220: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10221: { Writes the title of the graph }
10222: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10223: { Sets the maximum number of curves and re-initializes their parameters }
10224: procedure SetPointParam
10225: {$IFDEF DELPHI}
10226: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10227: {$ELSE}
10228: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10229: { Sets the point parameters for curve # CurvIndex }
10230: procedure SetLineParam
10231: {$IFDEF DELPHI}
```

```
(CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
{$ELSE}
(CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
{ Sets the line parameters for curve # CurvIndex }
procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
{ Sets the legend for curve # CurvIndex }
procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
{ Sets the step for curve # CurvIndex }
function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
procedure GetPointParam
{$IFDEF DELPHI}
(CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
{$ELSE}
(CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
{ Returns the point parameters for curve # CurvIndex }
procedure GetLineParam
{$IFDEF DELPHI}
(CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
{$ELSE}
(CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
{ Returns the line parameters for curve # CurvIndex }
function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
{ Returns the legend for curve # CurvIndex }
function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
{ Returns the step for curve # CurvIndex }
{$IFDEF DELPHI}
procedure PlotPoint(Canvas    : TCanvas;
                    X, Y      : Float; CurvIndex : Integer); external 'dmath';
{$ELSE}
procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
{$ENDIF}
{ Plots a point on the screen }
procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
                    X, Y                  : TVector;
                   Lb, Ub, CurvIndex      : Integer); external 'dmath';
{ Plots a curve }
procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
                                 X, Y, S               : TVector;
                                 Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
{ Plots a curve with error bars }
procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
                   Func                  : TFunc;
                   Xmin, Xmax            : Float;
                   {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
                   CurvIndex             : Integer); external 'dmath';
{ Plots a function }
procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
                      NCurv               : Integer;
                      ShowPoints, ShowLines : Boolean); external 'dmath';
{ Writes the legends for the plotted curves }
procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
                 Nx, Ny, Nc            : Integer;
                 X, Y, Z               : TVector;
                 F                     : TMatrix); external 'dmath';
{ Contour plot }
function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
{$IFNDEF DELPHI}
procedure LeaveGraphics; external 'dmath';
{ Quits graphic mode }
{$ENDIF}
{ ---------------------------------------------------------------------
  LaTeX graphics
  --------------------------------------------------------------------- }
function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
                          Header            : Boolean) : Boolean; external 'dmath';
{ Initializes the LaTeX file }
procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
{ Sets the graphic window }
procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
{ Sets the scale on the Ox axis }
procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
{ Sets the scale on the Oy axis }
procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
{ Sets the maximum number of curves and re-initializes their parameters }
procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
{ Sets the point parameters for curve # CurvIndex }
procedure TeX_SetLineParam(CurvIndex, Style : Integer;
                           Width : Float; Smooth : Boolean); external 'dmath';
```

```
10321: { Sets the line parameters for curve # CurvIndex }
10322: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10323: { Sets the legend for curve # CurvIndex }
10324: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10325: { Sets the step for curve # CurvIndex }
10326: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10327: { Plots a curve }
10328: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10329:                                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10330: { Plots a curve with error bars }
10331: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10332:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
10333: { Plots a function }
10334: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10335: { Writes the legends for the plotted curves }
10336: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
10337: { Contour plot }
10338: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10339: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10340:
10341: //****************************************************unit uPSI_SynPdf;
10342:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10343:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10344:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10345:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10346:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10347:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10348:  //Function _GetCharCount( Text : PAnsiChar) : integer
10349:  //Procedure L2R( W : PWideChar; L : integer)
10350:  Function PdfCoord( MM : single) : integer
10351:  Function CurrentPrinterPaperSize : TPDFPaperSize
10352:  Function CurrentPrinterRes : TPoint
10353:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10354:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10355:  Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10356:  Const('Usp10','String 'usp10.dll
10357:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10358:    'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10359:   AddTypeS('TScriptState_set', 'set of TScriptState_enum
10360: //*********************************************************************
10361:
10362: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10363: begin
10364:  Procedure PMrandomize( I : word)
10365:  Function PMrandom : longint
10366:  Function Rrand : extended
10367:  Function Irand( N : word) : word
10368:  Function Brand( P : extended) : boolean
10369:  Function Nrand : extended
10370: end;
10371:
10372: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10373: begin
10374:   Function Endian( x : LongWord) : LongWord
10375:   Function Endian64( x : Int64) : Int64
10376:   Function spRol( x : LongWord; y : Byte) : LongWord
10377:   Function spRor( x : LongWord; y : Byte) : LongWord
10378:   Function Ror64( x : Int64; y : Byte) : Int64
10379: end;
10380:
10381: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10382: begin
10383:  Procedure ClearModules
10384:  Procedure ReadMapFile( Fname : string)
10385:  Function AddressInfo( Address : dword) : string
10386: end;
10387:
10388: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10389: begin
10390:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10391:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10392:    +'teByOther, tpExecuteByOther )
10393:   TTarPermissions', 'set of TTarPermission
10394:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10395:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10396:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10397:   TTarModes', 'set of TTarMode
10398:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10399:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10400:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10401:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10402:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10403:   SIRegister_TTarArchive(CL);
10404:   SIRegister_TTarWriter(CL);
10405:  Function PermissionString( Permissions : TTarPermissions) : STRING
10406:  Function ConvertFilename( Filename : STRING) : STRING
10407:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10408:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10409:  Procedure ClearDirRec( var DirRec : TTarDirRec)
```

```
10410: end;
10411:
10412:
10413: //************************************************unit uPSI_TlHelp32;
10414: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10415: begin
10416:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10417:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10418:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10419:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10420:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10421:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10422:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10423:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10424:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10425:   AddTypeS('THeapList32', 'tagHEAPLIST32
10426:  Const('HF32_DEFAULT','LongInt'( 1);
10427:  Const('HF32_SHARED','LongInt'( 2);
10428:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10429:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10430:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10431:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10432:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10433:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10434:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10435:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10436:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10437:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10438:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10439:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10440:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10441:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10442:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10443:    +'aPri : Longint; dwFlags : DWORD; end
10444:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10445:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10446:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10447:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10448: end;
10449:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10450:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10451:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10452:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10453:  Const('EWX_LOGOFF','LongInt'( 0);
10454:  Const('EWX_SHUTDOWN','LongInt'( 1);
10455:  Const('EWX_REBOOT','LongInt'( 2);
10456:  Const('EWX_FORCE','LongInt'( 4);
10457:  Const('EWX_POWEROFF','LongInt'( 8);
10458:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10459:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10460:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10461:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10462:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10463:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10464:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10465:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10466:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10467:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10468:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10469:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10470:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10471:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10472:  Function GetDesktopWindow : HWND
10473:  Function GetParent( hWnd : HWND) : HWND
10474:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10475:  Function GetTopWindow( hWnd : HWND) : HWND
10476:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10477:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10478:  //Delphi DFM
10479:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10480:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10481:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10482:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10483:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10484:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10485:  Function OpenIcon( hWnd : HWND) : BOOL
10486:  Function CloseWindow( hWnd : HWND) : BOOL
10487:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10488:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10489:  Function IsWindowVisible( hWnd : HWND) : BOOL
10490:  Function IsIconic( hWnd : HWND) : BOOL
10491:  Function AnyPopup : BOOL
10492:  Function BringWindowToTop( hWnd : HWND) : BOOL
10493:  Function IsZoomed( hWnd : HWND) : BOOL
10494:  Function IsWindow( hWnd : HWND) : BOOL
10495:  Function IsMenu( hMenu : HMENU) : BOOL
10496:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10497:  Function DestroyWindow( hWnd : HWND) : BOOL
10498:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
```

```
10499:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10500:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10501:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10502:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10503:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10504:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10505:
10506: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10507: begin
10508:  const('ShowSetupDialogOptLong','String '--setup
10509:  PrimaryConfPathOptLong','String '--primary-config-path=
10510:  PrimaryConfPathOptShort','String '--pcp=
10511:  SecondaryConfPathOptLong','String '--secondary-config-path=
10512:  SecondaryConfPathOptShort','String '--scp=
10513:  NoSplashScreenOptLong','String '--no-splash-screen
10514:  NoSplashScreenOptShort','String '--nsc
10515:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10516:  SkipLastProjectOpt','String '--skip-last-project
10517:  DebugLogOpt','String '--debug-log=
10518:  DebugLogOptEnable','String '--debug-enable=
10519:  LanguageOpt','String '--language=
10520:  LazarusDirOpt','String '--lazarusdir=
10521:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10522:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10523:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10524:  Function IsHelpRequested : Boolean
10525:  Function IsVersionRequested : boolean
10526:  Function GetLanguageSpecified : string
10527:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10528:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10529:  Procedure ParseNoGuiCmdLineParams
10530:  Function ExtractCmdLineFilenames : TStrings
10531: end;
10532:
10533:
10534: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10535: begin
10536:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10537:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10538:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10539:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10540:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10541:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10542:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10543:  Function DirPathExists( DirectoryName : string) : boolean
10544:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10545:  Function ExtractFileNameOnly( const AFilename : string) : string
10546:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10547:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10548:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10549:  Function ForceDirectory( DirectoryName : string) : boolean
10550:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10551:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10552:  Function FileIsText( const AFilename : string) : boolean
10553:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10554:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10555:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10556:  Function TrimFilename( const AFilename : string) : string
10557:  Function ResolveDots( const AFilename : string) : string
10558:  Procedure ForcePathDelims( var FileName : string)
10559:  Function GetForcedPathDelims( const FileName : string) : String
10560:  Function CleanAndExpandFilename( const Filename : string) : string
10561:  Function CleanAndExpandDirectory( const Filename : string) : string
10562:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10563:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10564:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10565:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
10566:  Function FileIsInPath( const Filename, Path : string) : boolean
10567:  Function AppendPathDelim( const Path : string) : string
10568:  Function ChompPathDelim( const Path : string) : string
10569:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10570:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10571:  Function MinimizeSearchPath( const SearchPath : string) : string
10572:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10573:  (*Function FileExistsUTF8( const Filename : string) : boolean
10574:  Function FileAgeUTF8( const FileName : string) : Longint
10575:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10576:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10577:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10578:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10579:  Procedure FindCloseUTF8( var F : TSearchrec)
10580:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10581:  Function FileGetAttrUTF8( const FileName : String) : Longint
10582:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10583:  Function DeleteFileUTF8( const FileName : String) : Boolean
10584:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10585:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
```

```
10586:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10587:  Function GetCurrentDirUTF8 : String
10588:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10589:  Function CreateDirUTF8( const NewDir : String) : Boolean
10590:  Function RemoveDirUTF8( const Dir : String) : Boolean
10591:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10592:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10593:  Function FileCreateUTF8( const FileName : string) : THandle;
10594:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10595:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10596:  Function FileSizeUtf8( const Filename : string) : int64
10597:  Function GetFileDescription( const AFilename : string) : string
10598:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10599:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10600:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10601:  Function IsUNCPath( const Path : String) : Boolean
10602:  Function ExtractUNCVolume( const Path : String) : String
10603:  Function ExtractFileRoot( FileName : String) : String
10604:  Function GetDarwinSystemFilename( Filename : string) : string
10605:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10606:  Function StrToCmdLineParam( const Param : string) : string
10607:  Function MergeCmdLineParams( ParamList : TStrings) : string
10608:  Procedure InvalidateFileStateCache( const Filename : string)
10609:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10610:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10611:  Function FindAllDocs(const Root, extmask: string): TStringlist;
10612:  Function ReadFileToString( const Filename : string) : string
10613:  procedure Inc1(var X: longint; N: Longint);
10614:
10615:  type
10616:   TCopyFileFlag = ( cffOverwriteFile,
10617:     cffCreateDestDirectory, cffPreserveTime );
10618:   TCopyFileFlags = set of TCopyFileFlag;*)
10619:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10620:   TCopyFileFlags', 'set of TCopyFileFlag
10621:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10622: end;
10623:
10624: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10625: begin
10626:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10627:   SIRegister_TMask(CL);
10628:   SIRegister_TParseStringList(CL);
10629:   SIRegister_TMaskList(CL);
10630:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10631:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10632:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10633:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10634: end;
10635:
10636: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10637: begin
10638:   //PShellHookInfo', '^TShellHookInfo // will not work
10639:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10640:   SHELLHOOKINFO', 'TShellHookInfo
10641:   LPSHELLHOOKINFO', 'PShellHookInfo
10642:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10643:   SIRegister_TJvShellHook(CL);
10644:  Function InitJvShellHooks : Boolean
10645:  Procedure UnInitJvShellHooks
10646: end;
10647:
10648: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10649: begin
10650:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10651:    +', dcHasSetSel, dcWantTab, dcNative )
10652:   TDlgCodes', 'set of TDlgCode
10653:  'dcWantMessage',' dcWantAllKeys);
10654:   SIRegister_IJvExControl(CL);
10655:   SIRegister_IJvDenySubClassing(CL);
10656:   SIRegister_TStructPtrMessage(CL);
10657:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10658:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10659:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10660:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10661:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10662:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10663:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10664:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10665:  Function GetFocusedControl( AControl : TControl) : TWinControl
10666:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10667:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10668:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10669:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10670:   SIRegister_TJvExControl(CL);
10671:   SIRegister_TJvExWinControl(CL);
10672:   SIRegister_TJvExCustomControl(CL);
10673:   SIRegister_TJvExGraphicControl(CL);
10674:   SIRegister_TJvExHintWindow(CL);
```

```
10675:   SIRegister_TJvExPubGraphicControl(CL);
10676: end;
10677:
10678: (*----------------------------------------------------------------------------*)
10679: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10680: begin
10681:  Procedure EncodeStream( Input, Output : TStream)
10682:  Procedure DecodeStream( Input, Output : TStream)
10683:  Function EncodeString1( const Input : string) : string
10684:  Function DecodeString1( const Input : string) : string
10685: end;
10686:
10687: (*----------------------------------------------------------------------------*)
10688: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10689: begin
10690:   SIRegister_TWebAppRegInfo(CL);
10691:   SIRegister_TWebAppRegList(CL);
10692:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10693:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10694:  Procedure UnregisterWebApp( const AProgID : string)
10695:  Function FindRegisteredWebApp( const AProgID : string) : string
10696:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10697:  'sUDPPort','String 'UDPPort
10698: end;
10699:
10700: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10701: begin
10702:  // TStringDynArray', 'array of string
10703:  Function GetEnvVarValue( const VarName : string) : string
10704:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10705:  Function DeleteEnvVar( const VarName : string) : Integer
10706:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10707:  Function ExpandEnvVars( const Str : string) : string
10708:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10709:  Procedure GetAllEnvVarNames( const Names : TStrings);
10710:  Function GetAllEnvVarNames1 : TStringDynArray;
10711:  Function EnvBlockSize : Integer
10712:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10713:   SIRegister_TPJEnvVarsEnumerator(CL);
10714:   SIRegister_TPJEnvVars(CL);
10715:   FindClass('TOBJECT'),'EPJEnvVars
10716:   FindClass('TOBJECT'),'EPJEnvVars
10717:  //Procedure Register
10718: end;
10719:
10720: (*----------------------------------------------------------------------------*)
10721: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10722: begin
10723:  'cOneSecInMS','LongInt'( 1000);
10724:  //'cDefTimeSlice','LongInt'( 50);
10725:  //'cDefMaxExecTime',' cOneMinInMS);
10726:  'cAppErrorMask','LongInt'( 1 shl 29);
10727:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10728:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10729:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10730:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10731:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10732:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10733:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10734:   SIRegister_TPJCustomConsoleApp(CL);
10735:   SIRegister_TPJConsoleApp(CL);
10736: end;
10737:
10738: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10739: begin
10740:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10741:   t_encoding', '( uuencode, base64, mime )
10742:  Function internet_date( date : TDateTime) : string
10743:  Function lookup_hostname( const hostname : string) : longint
10744:  Function my_hostname : string
10745:  Function my_ip_address : longint
10746:  Function ip2string( ip_address : longint) : string
10747:  Function resolve_hostname( ip : longint) : string
10748:  Function address_from( const s : string; count : integer) : string
10749:  Function encode_base64( data : TStream) : TStringList
10750:  Function decode_base64( source : TStringList) : TMemoryStream
10751:  Function posn( const s, t : string; count : integer) : integer
10752:  Function poscn( c : char; const s : string; n : integer) : integer
10753:  Function filename_of( const s : string) : string
10754:  //Function trim( const s : string) : string
10755:  //Procedure setlength( var s : string; l : byte)
10756:  Function TimeZoneBias : longint
10757:  Function eight2seven_quoteprint( const s : string) : string
10758:  Function eight2seven_german( const s : string) : string
10759:  Function seven2eight_quoteprint( const s : string) : string end;
10760:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10761:  Function socketerror : cint
10762:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
```

```
10763:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10764:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10765:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10766:  Function fplisten( s : cint; backlog : cint) : cint
10767:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10768:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10769:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10770:  Function NetAddrToStr( Entry : in_addr) : String
10771:  Function HostAddrToStr( Entry : in_addr) : String
10772:  Function StrToHostAddr( IP : String) : in_addr
10773:  Function StrToNetAddr( IP : String) : in_addr
10774:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10775:   cint8', 'shortint
10776:   cuint8', 'byte
10777:   cchar', 'cint8
10778:   cschar', 'cint8
10779:   cuchar', 'cuint8
10780:   cint16', 'smallint
10781:   cuint16', 'word
10782:   cshort', 'cint16
10783:   csshort', 'cint16
10784:   cushort', 'cuint16
10785:   cint32', 'longint
10786:   cuint32', 'longword
10787:   cint', 'cint32
10788:   csint', 'cint32
10789:   cuint', 'cuint32
10790:   csigned', 'cint
10791:   cunsigned', 'cuint
10792:   cint64', 'int64
10793:   clonglong', 'cint64
10794:   cslonglong', 'cint64
10795:   cbool', 'longbool
10796:   cfloat', 'single
10797:   cdouble', 'double
10798:   clongdouble', 'extended
10799:
10800: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10801: begin
10802:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10803:   SIRegister_TlkJSONdotnetclass(CL);
10804:   SIRegister_TlkJSONbase(CL);
10805:   SIRegister_TlkJSONnumber(CL);
10806:   SIRegister_TlkJSONstring(CL);
10807:   SIRegister_TlkJSONboolean(CL);
10808:   SIRegister_TlkJSONnull(CL);
10809:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10810:    +'se; data : TObject; var Continue : Boolean)
10811:   SIRegister_TlkJSONcustomlist(CL);
10812:   SIRegister_TlkJSONlist(CL);
10813:   SIRegister_TlkJSONobjectmethod(CL);
10814:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10815:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10816:   SIRegister_TlkHashTable(CL);
10817:   SIRegister_TlkBalTree(CL);
10818:   SIRegister_TlkJSONobject(CL);
10819:   SIRegister_TlkJSON(CL);
10820:   SIRegister_TlkJSONstreamed(CL);
10821:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10822: end;
10823:
10824: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10825: begin
10826:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10827:   SIRegister_TZSortedList(CL);
10828:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10829:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10830:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10831:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10832:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10833:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10834:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10835:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10836:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10837:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10838:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10839:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10840:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10841:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10842:  Function StrToBoolEx( Str : string) : Boolean
10843:  Function BoolToStrEx( Bool : Boolean) : String
10844:  Function IsIpAddr( const Str : string) : Boolean
10845:  Function zSplitString( const Str, Delimiters : string) : TStrings
10846:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10847:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10848:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10849:  Function FloatToSQLStr( Value : Extended) : string
10850:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10851:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
```

```
10852:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10853:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10854:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10855:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10856:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10857:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10858:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10859:  Function BytesToVar( const Value : TByteDynArray) : Variant
10860:  Function VarToBytes( const Value : Variant) : TByteDynArray
10861:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10862:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10863:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10864:  Function EncodeCString( const Value : string) : string
10865:  Function DecodeCString( const Value : string) : string
10866:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10867:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10868:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
10869:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
       SubVersion:Integer):Int;
10870:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10871:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10872:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10873:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10874:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10875:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10876:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10877:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10878:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10879:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10880: end;
10881:
10882: unit uPSI_ZEncoding;
10883:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10884:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10885:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10886:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10887:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10888:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10889:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10890:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10891:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10892:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10893:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10894:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10895:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10896:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10897:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10898:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10899:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10900:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10901:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10902:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10903:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10904:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10905:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10906:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10907:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10908:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10909:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10910:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10911:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10912:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10913:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10914:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10915:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10916:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10917:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10918:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10919:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10920:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10921:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10922:  Function ZDefaultSystemCodePage : Word
10923:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10924:
10925:
10926: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10927: begin
10928:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10929:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10930:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10931:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10932:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10933:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10934:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10935:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10936:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10937:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10938:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
```

```
10939:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10940:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10941:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10942:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10943:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10944:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10945:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10946:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10947:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10948:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10949:  ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
10950:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10951:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10952:  ('EOAC_NONE','LongWord').SetUInt( $0);
10953:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10954:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10955:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10956:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10957:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10958:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10959:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10960:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10961:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10962:   FindClass('TOBJECT'),'EBoldCom
10963:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10964:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10965:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10966:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10967:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10968:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10969:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10970:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10971:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10972:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10973:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10974:  Function BoldCreateGUID : TGUID
10975:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
10976:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
10977:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10978:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10979: end;
10980:
10981: (*----------------------------------------------------------------------------*)
10982: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10983: begin
10984:  Function ParseISODate( s : string) : TDateTime
10985:  Function ParseISODateTime( s : string) : TDateTime
10986:  Function ParseISOTime( str : string) : TDateTime
10987: end;
10988:
10989: (*----------------------------------------------------------------------------*)
10990: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10991: begin
10992:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10993:  Function BoldCreateGUIDWithBracketsAsString : string
10994: end;
10995:
10996: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10997: begin
10998:   FindClass('TOBJECT'),'TBoldFileHandler
10999:   FindClass('TOBJECT'),'TBoldDiskFileHandler
11000:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11001:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
11002:   SIRegister_TBoldFileHandler(CL);
11003:   SIRegister_TBoldDiskFileHandler(CL);
11004:  Procedure BoldCloseAllFilehandlers
11005:  Procedure BoldRemoveUnchangedFilesFromEditor
11006:  Function BoldFileHandlerList : TBoldObjectArray
11007:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11008: end;
11009:
11010: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11011: begin
11012:   PCharArr', 'array of PChar
11013:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
11014:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11015:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
11016:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
11017:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
11018:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
11019:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
11020:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
```

```
11021:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11022:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11023:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
        Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11024:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11025: end;
11026:
11027: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11028: begin
11029:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11030:   SIRegister_TfrmBoldQueryUser(CL);
11031:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
11032: end;
11033:
11034: (*----------------------------------------------------------------------------*)
11035: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11036: begin
11037:  //('befIsInDisplayList',' BoldElementFlag0);
11038:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11039:  //('befFollowerSelected',' BoldElementFlag2);
11040:   FindClass('TOBJECT'),'TBoldQueue
11041:   FindClass('TOBJECT'),'TBoldQueueable
11042:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11043:   SIRegister_TBoldQueueable(CL);
11044:   SIRegister_TBoldQueue(CL);
11045:  Function BoldQueueFinalized : Boolean
11046:  Function BoldInstalledQueue : TBoldQueue
11047: end;
11048:
11049: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
11050: begin
11051:  const mmPerInch','Extended').setExtended( 25.4);
11052:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11053:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11054:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11055:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11056:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11057:   TBarLineType', '( white, black, black_half )
11058:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11059:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11060:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11061:   TCheckSumMethod', '( csmNone, csmModulo10 )
11062:   SIRegister_TAsBarcode(CL);
11063:  Function CheckSumModulo10( const data : string) : string
11064:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11065:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11066:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11067:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11068: end;
11069:
11070: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11071: begin
11072:    THomogeneousByteVector', 'array[0..3] of Byte
11073:   THomogeneousWordVector', 'array[0..3] of Word
11074:   THomogeneousIntVector', 'array[0..3] of Integer
11075:   THomogeneousFltVector', 'array[0..3] of single
11076:   THomogeneousDblVector', 'array[0..3] of double
11077:   THomogeneousExtVector', 'array[0..3] of extended
11078:   TAffineByteVector', 'array[0..2] of Byte
11079:   TAffineWordVector', 'array[0..2] of Word
11080:   TAffineIntVector', 'array[0..2] of Integer
11081:   TAffineFltVector', 'array[0..2] of single
11082:   TAffineDblVector', 'array[0..2] of double
11083:   TAffineExtVector', 'array[0..2] of extended
11084:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11085:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11086:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11087:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11088:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11089:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11090:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11091:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11092:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11093:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11094:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11095:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11096:   TMatrix4b', 'THomogeneousByteMatrix
11097:   TMatrix4w', 'THomogeneousWordMatrix
11098:   TMatrix4i', 'THomogeneousIntMatrix
11099:   TMatrix4f', 'THomogeneousFltMatrix
11100:   TMatrix4d', 'THomogeneousDblMatrix
11101:   TMatrix4e', 'THomogeneousExtMatrix
11102:   TMatrix3b', 'TAffineByteMatrix
11103:   TMatrix3w', 'TAffineWordMatrix
11104:   TMatrix3i', 'TAffineIntMatrix
11105:   TMatrix3f', 'TAffineFltMatrix
11106:   TMatrix3d', 'TAffineDblMatrix
11107:   TMatrix3e', 'TAffineExtMatrix
11108:   //'PMatrix', '^TMatrix // will not work
```

```
11109:   TMatrixGL', 'THomogeneousFltMatrix
11110:   THomogeneousMatrix', 'THomogeneousFltMatrix
11111:   TAffineMatrix', 'TAffineFltMatrix
11112:   TQuaternion', 'record Vector : TVector4f; end
11113:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11114:   +'ger; Height : Integer; end
11115:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11116:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11117:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11118:  'EPSILON','Extended').setExtended( 1E-100);
11119:  'EPSILON2','Extended').setExtended( 1E-50);
11120:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11121:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11122:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11123:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11124:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11125:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11126:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11127:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11128:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11129:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11130:  Function VectorLength( V : array of Single) : Single
11131:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11132:  Procedure VectorNegate( V : array of Single)
11133:  Function VectorNorm( V : array of Single) : Single
11134:  Function VectorNormalize( V : array of Single) : Single
11135:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11136:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11137:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11138:  Procedure VectorScale( V : array of Single; Factor : Single)
11139:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11140:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11141:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11142:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11143:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11144:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11145:  Procedure MatrixAdjoint( var M : TMatrixGL)
11146:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11147:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11148:  Function MatrixDeterminant( M : TMatrixGL) : Single
11149:  Procedure MatrixInvert( var M : TMatrixGL)
11150:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11151:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11152:  Procedure MatrixTranspose( var M : TMatrixGL)
11153:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11154:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11155:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11156:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11157:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11158:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11159:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11160:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11161:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11162:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11163:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11164:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11165:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11166:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11167:  Function MakeAffineVector( V : array of Single) : TAffineVector
11168:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11169:  Function MakeVector( V : array of Single) : TVectorGL
11170:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11171:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11172:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11173:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11174:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11175:  Function ArcCosGL( X : Extended) : Extended
11176:  Function ArcSinGL( X : Extended) : Extended
11177:  Function ArcTan2GL( Y, X : Extended) : Extended
11178:  Function CoTanGL( X : Extended) : Extended
11179:  Function DegToRadGL( Degrees : Extended) : Extended
11180:  Function RadToDegGL( Radians : Extended) : Extended
11181:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11182:  Function TanGL( X : Extended) : Extended
11183:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11184:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11185:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11186:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11187:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11188:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11189: end;
11190:
11191:
11192: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11193: begin
11194:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11195:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11196:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11197:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
```

```
11198:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11199:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11200:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11201:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11202:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11203:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11204:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11205:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11206:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11207:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11208:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11209:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11210:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11211:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11212:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11213:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11214:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11215:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11216:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11217:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11218:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11219:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
       SaveTo:TStrings):Bool;
11220:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11221: end;
11222:
11223: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11224: begin
11225:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11226:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11227:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11228:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11229:   FindClass('TOBJECT'),'EInvalidParam
11230:  Function IsDCOMInstalled : Boolean
11231:  Function IsDCOMEnabled : Boolean
11232:  Function GetDCOMVersion : string
11233:  Function GetMDACVersion : string
11234:  Function GetMDACVersion2 : string
11235:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11236:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11237:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11238:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11239:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11240:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11241:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11242:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11243:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11244:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11245:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11246:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11247:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11248:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11249: end;
11250:
11251:
11252: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11253: begin
11254:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11255:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11256:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11257:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11258:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11259:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11260:  DegPerCycle','Extended').setExtended( 360.0);
11261:  DegPerGrad','Extended').setExtended( 0.9);
11262:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11263:  GradPerCycle','Extended').setExtended( 400.0);
11264:  GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
11265:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11266:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11267:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11268:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11269:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11270:  CyclePerGrad','Extended').setExtended( 0.0025);
11271:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11272:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11273:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11274:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11275:  Function MakePercentage( const Step, Max : Longint) : Longint
11276:  Function CelsiusToKelvin( const T : double) : double
11277:  Function CelsiusToFahrenheit( const T : double) : double
11278:  Function KelvinToCelsius( const T : double) : double
11279:  Function KelvinToFahrenheit( const T : double) : double
11280:  Function FahrenheitToCelsius( const T : double) : double
11281:  Function FahrenheitToKelvin( const T : double) : double
```

```
11282:  Function CycleToDeg( const Cycles : double) : double
11283:  Function CycleToGrad( const Cycles : double) : double
11284:  Function CycleToRad( const Cycles : double) : double
11285:  Function DegToCycle( const Degrees : double) : double
11286:  Function DegToGrad( const Degrees : double) : double
11287:  Function DegToRad( const Degrees : double) : double
11288:  Function GradToCycle( const Grads : double) : double
11289:  Function GradToDeg( const Grads : double) : double
11290:  Function GradToRad( const Grads : double) : double
11291:  Function RadToCycle( const Radians : double) : double
11292:  Function RadToDeg( const Radians : double) : double
11293:  Function RadToGrad( const Radians : double) : double
11294:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11295:  Function DmsToRad( const D, M : Integer; const S : double) : double
11296:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11297:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11298:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11299:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11300:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11301:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11302:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11303:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11304:  Function CmToInch( const Cm : double) : double
11305:  Function InchToCm( const Inch : double) : double
11306:  Function FeetToMetre( const Feet : double) : double
11307:  Function MetreToFeet( const Metre : double) : double
11308:  Function YardToMetre( const Yard : double) : double
11309:  Function MetreToYard( const Metre : double) : double
11310:  Function NmToKm( const Nm : double) : double
11311:  Function KmToNm( const Km : double) : double
11312:  Function KmToSm( const Km : double) : double
11313:  Function SmToKm( const Sm : double) : double
11314:  Function LitreToGalUs( const Litre : double) : double
11315:  Function GalUsToLitre( const GalUs : double) : double
11316:  Function GalUsToGalCan( const GalUs : double) : double
11317:  Function GalCanToGalUs( const GalCan : double) : double
11318:  Function GalUsToGalUk( const GalUs : double) : double
11319:  Function GalUkToGalUs( const GalUk : double) : double
11320:  Function LitreToGalCan( const Litre : double) : double
11321:  Function GalCanToLitre( const GalCan : double) : double
11322:  Function LitreToGalUk( const Litre : double) : double
11323:  Function GalUkToLitre( const GalUk : double) : double
11324:  Function KgToLb( const Kg : double) : double
11325:  Function LbToKg( const Lb : double) : double
11326:  Function KgToOz( const Kg : double) : double
11327:  Function OzToKg( const Oz : double) : double
11328:  Function CwtUsToKg( const Cwt : double) : double
11329:  Function CwtUkToKg( const Cwt : double) : double
11330:  Function KaratToKg( const Karat : double) : double
11331:  Function KgToCwtUs( const Kg : double) : double
11332:  Function KgToCwtUk( const Kg : double) : double
11333:  Function KgToKarat( const Kg : double) : double
11334:  Function KgToSton( const Kg : double) : double
11335:  Function KgToLton( const Kg : double) : double
11336:  Function StonToKg( const STon : double) : double
11337:  Function LtonToKg( const Lton : double) : double
11338:  Function QrUsToKg( const Qr : double) : double
11339:  Function QrUkToKg( const Qr : double) : double
11340:  Function KgToQrUs( const Kg : double) : double
11341:  Function KgToQrUk( const Kg : double) : double
11342:  Function PascalToBar( const Pa : double) : double
11343:  Function PascalToAt( const Pa : double) : double
11344:  Function PascalToTorr( const Pa : double) : double
11345:  Function BarToPascal( const Bar : double) : double
11346:  Function AtToPascal( const At : double) : double
11347:  Function TorrToPascal( const Torr : double) : double
11348:  Function KnotToMs( const Knot : double) : double
11349:  Function HpElectricToWatt( const HpE : double) : double
11350:  Function HpMetricToWatt( const HpM : double) : double
11351:  Function MsToKnot( const ms : double) : double
11352:  Function WattToHpElectric( const W : double) : double
11353:  Function WattToHpMetric( const W : double) : double
11354:  function getBigPI: string;    //PI of 1000 numbers
11355:
11356: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11357: begin
11358:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11359:  Procedure CDCopyFile( const FileName, DestName : string)
11360:  Procedure CDMoveFile( const FileName, DestName : string)
11361:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11362:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11363:  Function CDGetTempDir : string
11364:  Function CDGetFileSize( FileName : string) : longint
11365:  Function GetFileTime( FileName : string) : longint
11366:  Function GetShortName( FileName : string) : string
11367:  Function GetFullName( FileName : string) : string
11368:  Function WinReboot : boolean
11369:  Function WinDir : String
11370:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
```

```
11371:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11372:  Function devExecutor : TdevExecutor
11373: end;
11374:
11375: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11376: begin
11377:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11378:  Procedure Associate( Index : integer)
11379:  Procedure UnAssociate( Index : integer)
11380:  Function IsAssociated( Index : integer) : boolean
11381:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11382:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11383:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11384:  procedure RefreshIcons;
11385: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11386: function MergColor(Colors: Array of TColor): TColor;
11387: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11388: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11389: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11390: function GetInverseColor(AColor: TColor): TColor;
11391: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11392: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11393: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11394: Procedure GetSystemMenuFont(Font: TFont);
11395: end;
11396:
11397: //*************************unit uPSI_JvHLParser;*****************************
11398: function IsStringConstant(const St: string): Boolean;
11399: function IsIntConstant(const St: string): Boolean;
11400: function IsRealConstant(const St: string): Boolean;
11401: function IsIdentifier(const ID: string): Boolean;
11402: function GetStringValue(const St: string): string;
11403: procedure ParseString(const S: string; Ss: TStrings);
11404: function IsStringConstantW(const St: WideString): Boolean;
11405: function IsIntConstantW(const St: WideString): Boolean;
11406: function IsRealConstantW(const St: WideString): Boolean;
11407: function IsIdentifierW(const ID: WideString): Boolean;
11408: function GetStringValueW(const St: WideString): WideString;
11409: procedure ParseStringW(const S: WideString; Ss: TStrings);
11410:
11411:
11412: //*************************unit uPSI_JclMapi;*****************************
11413:
11414: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11415: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11416: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11417: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11418: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11419:
11420: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11421: begin
11422:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11423:  Function BuildType1Message( ADomain, AHost : String) : String
11424:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11425:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11426:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11427: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11428: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11429: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11430: end;
11431:
11432: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11433: begin
11434: ('IpAny','LongWord').SetUInt( $00000000);
11435:  IpLoopBack','LongWord').SetUInt( $7F000001);
11436:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11437:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11438:  PortAny','LongWord( $0000);
11439:  SocketMaxConnections','LongInt'( 5);
11440:  TIpAddr', 'LongWord
11441:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11442:  Function HostToNetLong( HostLong : LongWord) : LongWord
11443:  Function HostToNetShort( HostShort : Word) : Word
11444:  Function NetToHostLong( NetLong : LongWord) : LongWord
11445:  Function NetToHostShort( NetShort : Word) : Word
11446:  Function StrToIp( Ip : string) : TIpAddr
11447:  Function IpToStr( Ip : TIpAddr) : string
11448: end;
11449:
11450: (*----------------------------------------------------------------------------*)
11451: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11452: begin
11453:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11454:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11455:    +'amSha1, AlsmtpClientAuthAutoSelect )
11456:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
```

```
11457:   SIRegister_TAlSmtpClient(CL);
11458: end;
11459:
11460: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11461: begin
11462: 'TBitNo', 'Integer
11463:   TStByteNo', 'Integer
11464: TStationNo', 'Integer
11465: TInOutNo', 'Integer
11466: TIo', '( EE, AA, NE, NA )
11467: TBitSet', 'set of TBitNo
11468: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11469: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11470: TBitAddr', 'LongInt
11471: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11472: TByteAddr', 'SmallInt
11473: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11474:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11475:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11476:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11477:  Function BitAddrToStr( Value : TBitAddr) : string
11478:  Function StrToBitAddr( const Value : string) : TBitAddr
11479:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11480:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11481:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11482:  Function ByteAddrToStr( Value : TByteAddr) : string
11483:  Function StrToByteAddr( const Value : string) : TByteAddr
11484:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11485:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11486:  Function InOutStateToStr( State : TInOutState) : string
11487:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11488:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11489: end;
11490:
11491: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11492: begin
11493: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11494:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11495: DpmiPmVector', 'Int64
11496:  'DInterval','LongInt'( 1000);
11497:  //'DEnabled','Boolean')BoolToStr( True);
11498:  'DIntFreq','string' if64
11499:  //'DMessages','Boolean if64);
11500:   SIRegister_TwdxCustomTimer(CL);
11501:   SIRegister_TwdxTimer(CL);
11502:   SIRegister_TwdxRtcTimer(CL);
11503:   SIRegister_TCustomIntTimer(CL);
11504:   SIRegister_TIntTimer(CL);
11505:   SIRegister_TRtcIntTimer(CL);
11506:  Function RealNow : TDateTime
11507:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11508:  Function DateTimeToMs( Time : TDateTime) : LongInt
11509: end;
11510:
11511: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11512: begin
11513: TIdSyslogPRI', 'Integer
11514: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11515:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11516:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11517:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11518:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11519: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11520:   SIRegister_TIdSysLogMsgPart(CL);
11521:   SIRegister_TIdSysLogMessage(CL);
11522:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11523:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11524:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11525:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11526:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11527:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11528: end;
11529:
11530: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11531: begin
11532:  'UWhitespace','String '(?:\s*)
11533:  Function StripSpaces( const AText : string) : string
11534:  Function CharCount( const AText : string; Ch : Char) : Integer
11535:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11536:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11537: end;
11538:
11539:
11540: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11541: begin
11542:  ExtPascalVersion','String '0.9.8
11543:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11544:    +'Opera, brKonqueror, brMobileSafari )
```

```
11545:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11546:   AddTypeS('TExtProcedure', 'Procedure
11547:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11548:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11549:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11550:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11551:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11552:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11553:  Function StrToJS( const S : string; UseBR : boolean) : string
11554:  Function CaseOf( const S : string; const Cases : array of string) : integer
11555:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11556:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11557:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11558:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11559:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11560:  Function IsUpperCase( S : string) : boolean
11561:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11562:  Function BeautifyCSS( const AStyle : string) : string
11563:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11564:  Function JSDateToDateTime( JSDate : string) : TDateTime
11565: end;
11566:
11567: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11568: begin
11569:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11570:   TSHDeleteOptions', 'set of TSHDeleteOption
11571:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11572:   TSHRenameOptions', 'set of TSHRenameOption
11573:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11574:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11575:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11576:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11577:   TEnumFolderFlags', 'set of TEnumFolderFlag
11578:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11579:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11580:    +'IEnumIdList; Folder : IShellFolder; end
11581:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11582:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11583:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11584:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11585:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11586:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11587:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11588:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11589:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11590:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11591:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11592:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11593:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11594:  Function SHFreeMem( var P : Pointer) : Boolean
11595:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11596:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11597:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11598:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11599:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11600:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11601:  Function PidlFree( var IdList : PItemIdList) : Boolean
11602:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11603:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11604:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11605:  Function PidlToPath( IdList : PItemIdList) : string
11606:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11607:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11608:   PShellLink', '^TShellLink // will not work
11609:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11610:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11611:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11612:  Procedure ShellLinkFree( var Link : TShellLink)
11613:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11614:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11615:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11616:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11617:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11618:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11619:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11620:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11621:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11622:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11623:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11624:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11625:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11626:  Function ShellOpenAs( const FileName : string) : Boolean
11627:  Function ShellRasDial( const EntryName : string) : Boolean
11628:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11629:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11630:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11631:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11632:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11633:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
```

```
11634:  Function OemKeyScan( wOemChar : Word) : DWORD
11635:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11636: end;
11637:
11638: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11639: begin
11640:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11641:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11642:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11643:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11644:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11645:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11646:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11647:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11648:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11649:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11650:  Function xmlValidName( const Text : UnicodeString) : Boolean
11651:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11652:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11653:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11654:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11655:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11656:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11657:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11658:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11659:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11660:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11661:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11662:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11663:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11664:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11665:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11666:  Procedure SelfTestcXMLFunctions
11667: end;
11668:
11669: (*----------------------------------------------------------------------------*)
11670: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11671: begin
11672:  Function AWaitCursor : IUnknown
11673:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11674:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11675:  Function YesNo( const ACaption, AMsg : string) : boolean
11676:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11677:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11678:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11679:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11680:  Procedure GetSystemPaths( Strings : TStrings)
11681:  Procedure MakeEditNumeric( EditHandle : integer)
11682: end;
11683:
11684: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11685: begin
11686:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11687:  'BI_YUY2','LongWord( $32595559);
11688:  'BI_UYVY','LongWord').SetUInt( $59565955);
11689:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11690:  'BI_YVU9','LongWord').SetUInt( $39555659);
11691:  'BI_YUV12','LongWord( $30323449);
11692:  'BI_Y8','LongWord').SetUInt( $20203859);
11693:  'BI_Y211','LongWord').SetUInt( $31313259);
11694:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11695:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11696: end;
11697:
11698: (*----------------------------------------------------------------------------*)
11699: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11700: begin
11701:  'WM_USER','LongWord').SetUInt( $0400);
11702:  'WM_CAP_START','LongWord').SetUint($0400);
11703:  'WM_CAP_END','longword').SetUint($0400+85);
11704:  //WM_CAP_START+  85
11705:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11706:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11707:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11708:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11709:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11710:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11711:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11712:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11713:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11714:  Function capGetUserData( hwnd : THandle) : LongInt
11715:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11716:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11717:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11718:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11719:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11720:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11721:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
```

```
11722:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11723:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11724:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11725:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11726:  Function capEditCopy( hwnd : THandle) : LongInt
11727:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11728:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11729:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11730:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11731:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11732:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11733:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11734:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11735:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11736:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11737:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11738:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11739:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11740:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11741:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11742:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11743:  Function capGrabFrame( hwnd : THandle) : LongInt
11744:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11745:  Function capCaptureSequence( hwnd : THandle) : LongInt
11746:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11747:  Function capCaptureStop( hwnd : THandle) : LongInt
11748:  Function capCaptureAbort( hwnd : THandle) : LongInt
11749:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11750:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11751:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11752:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11753:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11754:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11755:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11756:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11757:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11758:  Function capPalettePaste( hwnd : THandle) : LongInt
11759:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11760:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11761:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11762:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11763:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11764:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11765:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11766:   //PCapStatus', '^TCapStatus // will not work
11767:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11768:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11769:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11770:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11771:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11772:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11773:    +' wNumAudioAllocated : WORD; end
11774:   //PCaptureParms', '^TCaptureParms // will not work
11775:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11776:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11777:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11778:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11779:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11780:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11781:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11782:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11783:    +'he : BOOL; AVStreamMaster : WORD; end
11784:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11785:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11786:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11787:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11788:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
      : Integer; hwndParent : THandle; nID : Integer) : THandle
11789:  Function
      capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11790:  'IDS_CAP_BEGIN','LongInt'( 300);
11791:  'IDS_CAP_END','LongInt'( 301);
11792:  'IDS_CAP_INFO','LongInt'( 401);
11793:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11794:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11795:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11796:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11797:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11798:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11799:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11800:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11801:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11802:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11803:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11804:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11805:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11806:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11807:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11808:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
```

```
11809:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11810:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11811:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11812:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11813:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11814:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11815:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11816:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11817:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11818:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11819:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11820:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11821:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11822:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11823:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11824:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11825:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11826:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11827:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11828:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11829:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11830:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11831:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11832:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11833:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11834:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11835:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11836:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11837:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11838:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11839:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11840:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11841:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11842:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11843:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11844:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11845:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11846:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11847:  'AVICAP32','String 'AVICAP32.dll
11848: end;
11849:
11850: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11851: begin
11852:  Function AlBoolToInt( Value : Boolean) : Integer
11853:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11854:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11855:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11856:  Function ALInc( var x : integer; Count : integer) : Integer
11857:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11858:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11859:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11860:  Function ALIsInteger(const S: AnsiString): Boolean;
11861:  function ALIsDecimal(const S: AnsiString): boolean;
11862:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11863:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11864:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11865:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11866:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11867:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11868:  Function ALRandomStr(const aLength: Longint): AnsiString;
11869:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11870:  Function ALRandomStrU(const aLength: Longint): String;
11871: end;
11872:
11873: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11874: begin
11875:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
       aTrueStr: AnsiString; const aFalseStr : AnsiString)
11876: end;
11877:
11878: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11879: begin
11880:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11881:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11882:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11883:    +'; ullAvailExtendedVirtual : Int64; end
11884:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11885:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11886:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11887:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11888:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11889:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11890:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11891:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11892: end;
11893:
11894: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11895: begin
11896:   SIRegister_THandledObject(CL);
```

```
11897:   SIRegister_TEvent(CL);
11898:   SIRegister_TMutex(CL);
11899:   SIRegister_TSharedMem(CL);
11900:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11901:  'TRACE_BUFFER','String 'TRACE_BUFFER
11902:  'TRACE_MUTEX','String 'TRACE_MUTEX
11903:   //PTraceEntry', '^TTraceEntry // will not work
11904:   SIRegister_TIPCTracer(CL);
11905:  'MAX_CLIENTS','LongInt'( 6);
11906:  'IPCTIMEOUT','LongInt'( 2000);
11907:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11908:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11909:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11910:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11911:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11912:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11913:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11914:   FindClass('TOBJECT'),'EMonitorActive
11915:   FindClass('TOBJECT'),'TIPCThread
11916:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11917:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11918:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11919:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11920:   TClientFlags', 'set of TClientFlag
11921:   //PEventData', '^TEventData // will not work
11922:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11923:    +'lag; Flags : TClientFlags; end
11924:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11925:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11926:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11927:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11928:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11929:   SIRegister_TIPCEvent(CL);
11930:   //PClientDirRecords', '^TClientDirRecords // will not work
11931:   SIRegister_TClientDirectory(CL);
11932:   TIPCState', '( stInActive, stDisconnected, stConnected )
11933:   SIRegister_TIPCThread(CL);
11934:   SIRegister_TIPCMonitor(CL);
11935:   SIRegister_TIPCClient(CL);
11936:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11937: end;
11938:
11939: (*----------------------------------------------------------------------------*)
11940: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11941: begin
11942:   SIRegister_TAlGSMComm(CL);
11943:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11944:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
      AMessage:AnsiString);
11945:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11946:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
      UseGreekAlphabet:Bool):Widestring;
11947:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11948:  end;
11949:
11950: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11951: begin
11952:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11953:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11954:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11955:   TInternetScheme', 'integer
11956:   TALIPv6Binary', 'array[1..16] of Char;
11957:  // TALIPv6Binary = array[1..16] of ansiChar;
11958:  //   TInternetScheme = Integer;
11959:   SIRegister_TALHTTPRequestHeader(CL);
11960:   SIRegister_TALHTTPCookie(CL);
11961:   SIRegister_TALHTTPCookieCollection(CL);
11962:   SIRegister_TALHTTPResponseHeader(CL);
11963:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11964:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11965: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
      Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11966: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
      Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11967: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
      Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11968:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11969:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11970:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11971:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11972:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11973:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
      ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11974:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
      Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11975:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11976:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11977:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11978:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
```

```
11979:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11980:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11981:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11982:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11983:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11984:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11985:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
11986:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11987:  Function ALZeroIpV6 : TALIPv6Binary
11988:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11989:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
11990:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11991:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
11992: end;
11993:
11994: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11995: begin
11996:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
11997:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11998:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
11999:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
12000:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
12001:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
12002:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
12003:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
12004:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
12005:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12006:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
12007: end;
12008:
12009: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12010: begin
12011:   SIRegister_TALEMailHeader(CL);
12012:   SIRegister_TALNewsArticleHeader(CL);
12013:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
12014:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
12015:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
12016:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
12017:  Function AlGenerateInternetMessageID : AnsiString;
12018:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
12019:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
12020:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12021: end;
12022:
12023: (*----------------------------------------------------------------------------*)
12024: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12025: begin
12026:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12027:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
12028:  Function ALgetLocalIPs : TALStrings
12029:  Function ALgetLocalHostName : AnsiString
12030: end;
12031:
12032: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12033: begin
12034:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
12035:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12036:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12037:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
12038:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12039:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12040:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
12041:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12042:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
12043: end;
12044:
12045: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12046: begin
12047:   TStartupInfoA', 'TStartupInfo
12048:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12049:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12050:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12051:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12052:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12053:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
```

```
12054:  SE_TCB_NAME','String 'SeTcbPrivilege
12055:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12056:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12057:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12058:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12059:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12060:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12061:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12062:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12063:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12064:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12065:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12066:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12067:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12068:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12069:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12070:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12071:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12072:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12073:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12074:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12075:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12076:  Function AlGetEnvironmentString : AnsiString
12077:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12078:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12079:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12080:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12081:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12082: end;
12083:
12084: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12085: begin
12086:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12087:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12088:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12089:  Function ALGetModuleName : ansistring
12090:  Function ALGetModuleFileNameWithoutExtension : ansistring
12091:  Function ALGetModulePath : ansiString
12092:  Function AlGetFileSize( const AFileName : ansistring) : int64
12093:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12094:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12095:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12096:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12097:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12098:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12099:  Function ALFileExists( const Path : ansiString) : boolean
12100:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12101:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12102:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12103:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12104:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12105: end;
12106:
12107: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12108: begin
12109:   NativeInt', 'Integer
12110:   NativeUInt', 'Cardinal
12111:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12112:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12113:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12114:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12115:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12116:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12117:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12118:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12119:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12120:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12121:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12122:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12123:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12124:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12125:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12126:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12127:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12128:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
```

```
12129:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12130:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12131:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12132:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12133:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12134:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12135:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12136:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12137:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12138:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12139:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12140:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12141:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12142:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12143: end;
12144:
12145: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12146: begin
12147:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12148:   FindClass('TOBJECT'),'TALXMLNode
12149:   FindClass('TOBJECT'),'TALXMLNodeList
12150:   FindClass('TOBJECT'),'TALXMLDocument
12151:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12152:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12153:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12154:    +'nst Name : AnsiString; const Attributes : TALStrings)
12155:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12156:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12157:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12158:    +'ntDocType, ntDocFragment, ntNotation )
12159:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12160:   TALXMLDocOptions', 'set of TALXMLDocOption
12161:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12162:   TALXMLParseOptions', 'set of TALXMLParseOption
12163:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12164:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12165:   SIRegister_EALXMLDocError(CL);
12166:   SIRegister_TALXMLNodeList(CL);
12167:   SIRegister_TALXMLNode(CL);
12168:   SIRegister_TALXmlElementNode(CL);
12169:   SIRegister_TALXmlAttributeNode(CL);
12170:   SIRegister_TALXmlTextNode(CL);
12171:   SIRegister_TALXmlDocumentNode(CL);
12172:   SIRegister_TALXmlCommentNode(CL);
12173:   SIRegister_TALXmlProcessingInstrNode(CL);
12174:   SIRegister_TALXmlCDataNode(CL);
12175:   SIRegister_TALXmlEntityRefNode(CL);
12176:   SIRegister_TALXmlEntityNode(CL);
12177:   SIRegister_TALXmlDocTypeNode(CL);
12178:   SIRegister_TALXmlDocFragmentNode(CL);
12179:   SIRegister_TALXmlNotationNode(CL);
12180:   SIRegister_TALXMLDocument(CL);
12181:  cAlXMLUTF8EncodingStr','String 'UTF-8
12182:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12183:  CALNSDelim','String ':
12184:  CALXML','String 'xml
12185:  CALVersion','String 'version
12186:  CALEncoding','String 'encoding
12187:  CALStandalone','String 'standalone
12188:  CALDefaultNodeIndent','String '
12189:  CALXmlDocument','String 'DOCUMENT
12190:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12191:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
       EncodingStr:AnsiString);
12192:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
       ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12193:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
       ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12194:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
       Recurse: Boolean):TalxmlNode
12195:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
       AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12196:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
       AnsiString
12197: end;
12198:
12199: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12200: //based on TEEProc, TeCanvas, TEEngine, TChart
12201: begin
12202:  'TeePiStep','Double').setExtended( Pi / 180.0);
12203:  'TeeDefaultPerspective','LongInt'( 100);
12204:  'TeeMinAngle','LongInt'( 270);
12205:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12206:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12207:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12208:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12209:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12210:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
```

```
12211:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12212:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12213:  'TA_LEFT','LongInt'( 0);
12214:  'TA_RIGHT','LongInt'( 2);
12215:  'TA_CENTER','LongInt'( 6);
12216:  'TA_TOP','LongInt'( 0);
12217:  'TA_BOTTOM','LongInt'( 8);
12218:  'teePATCOPY','LongInt'( 0);
12219:  'NumCirclePoints','LongInt'( 64);
12220:  'teeDEFAULT_CHARSET','LongInt'( 1);
12221:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12222:  'TA_LEFT','LongInt'( 0);
12223:  'bs_Solid','LongInt'( 0);
12224:  'teepf24Bit','LongInt'( 0);
12225:  'teepfDevice','LongInt'( 1);
12226:  'CM_MOUSELEAVE','LongInt'( 10000);
12227:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12228:  'DC_BRUSH','LongInt'( 18);
12229:  'DC_PEN','LongInt'( 19);
12230:   teeCOLORREF', 'LongWord
12231:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12232:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12233:   SIRegister_TFilterRegion(CL);
12234:   SIRegister_IFormCreator(CL);
12235:   SIRegister_TTeeFilter(CL);
12236:   //TFilterClass', 'class of TTeeFilter
12237:   SIRegister_TFilterItems(CL);
12238:   SIRegister_TConvolveFilter(CL);
12239:   SIRegister_TBlurFilter(CL);
12240:   SIRegister_TTeePicture(CL);
12241:   TPenEndStyle', '( esRound, esSquare, esFlat )
12242:   SIRegister_TChartPen(CL);
12243:   SIRegister_TChartHiddenPen(CL);
12244:   SIRegister_TDottedGrayPen(CL);
12245:   SIRegister_TDarkGrayPen(CL);
12246:   SIRegister_TWhitePen(CL);
12247:   SIRegister_TChartBrush(CL);
12248:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12249:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12250:   SIRegister_TView3DOptions(CL);
12251:   FindClass('TOBJECT'),'TTeeCanvas
12252:   TTeeTransparency', 'Integer
12253:   SIRegister_TTeeBlend(CL);
12254:   FindClass('TOBJECT'),'TCanvas3D
12255:   SIRegister_TTeeShadow(CL);
12256:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
        gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12257:   FindClass('TOBJECT'),'TSubGradient
12258:   SIRegister_TCustomTeeGradient(CL);
12259:   SIRegister_TSubGradient(CL);
12260:   SIRegister_TTeeGradient(CL);
12261:   SIRegister_TTeeFontGradient(CL);
12262:   SIRegister_TTeeFont(CL);
12263:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12264:   TCanvasTextAlign', 'Integer
12265:   TTeeCanvasHandle', 'HDC
12266:   SIRegister_TTeeCanvas(CL);
12267:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12268:   SIRegister_TFloatXYZ(CL);
12269:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12270:   TRGB', 'record blue: byte; green: byte; red: byte; end
12271:   {TRGB=packed record
12272:     Blue  : Byte;
12273:     Green : Byte;
12274:     Red   : Byte;
12275:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12276:
12277: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12278:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12279: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12280: TCanvas3DPlane', '( cpX, cpY, cpZ )
12281:   //IInterface', 'IUnknown
12282:   SIRegister_TCanvas3D(CL);
12283:   SIRegister_TTeeCanvas3D(CL);
12284:   TTrianglePoints', 'Array[0..2] of TPoint;
12285:   TFourPoints', 'Array[0..3] of TPoint;
12286:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12287:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12288:  Function Point3D( const x, y, z : Integer) : TPoint3D
12289:  Procedure SwapDouble( var a, b : Double)
12290:  Procedure SwapInteger( var a, b : Integer)
12291:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12292:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12293:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12294:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12295:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12296:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12297:  Procedure UnClipCanvas( ACanvas : TCanvas)
12298:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
```

```
12299:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12300:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12301:  'TeeCharForHeight','String 'W
12302:  'DarkerColorQuantity','Byte').SetUInt( 128);
12303:  'DarkColorQuantity','Byte').SetUInt( 64);
12304:   TButtonGetColorProc', 'Function  : TColor
12305:   SIRegister_TTeeButton(CL);
12306:   SIRegister_TButtonColor(CL);
12307:   SIRegister_TComboFlat(CL);
12308:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12309:  Function TeePoint( const aX, aY : Integer) : TPoint
12310:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12311:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12312:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12313:  Function OrientRectangle( const R : TRect) : TRect
12314:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12315:  Function PolygonBounds( const P : array of TPoint) : TRect
12316:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12317:  Function RGBValue( const Color : TColor) : TRGB
12318:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12319:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12320:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12321:  Function TeeCull( const P : TFourPoints) : Boolean;
12322:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12323:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12324:  Procedure SmoothStretch( Src, Dst : TBitmap);
12325:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12326:  Function TeeDistance( const x, y : Double) : Double
12327:  Function TeeLoadLibrary( const FileName : String) : HInst
12328:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12329:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12330:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12331:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
      Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12332:   SIRegister_ICanvasHyperlinks(CL);
12333:   SIRegister_ICanvasToolTips(CL);
12334:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12335: end;
12336:
12337: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12338: begin
12339:   TOvcHdc', 'Integer
12340:   TOvcHWND', 'Cardinal
12341:   TOvcHdc', 'HDC
12342:   TOvcHWND', 'HWND
12343:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12344:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12345:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12346:  Function DefaultEpoch : Integer
12347:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12348:  Procedure FixRealPrim( P : PChar; DC : Char)
12349:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12350:  Function GetLeftButton : Byte
12351:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12352:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12353:  Function GetShiftFlags : Byte
12354:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12355:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12356:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12357:  Function ovIsForegroundTask : Boolean
12358:  Function ovTrimLeft( const S : string) : string
12359:  Function ovTrimRight( const S : string) : string
12360:  Function ovQuotedStr( const S : string) : string
12361:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12362:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12363:  Function PtrDiff( const P1, P2 : PChar) : Word
12364:  Procedure PtrInc( var P, Delta : Word)
12365:  Procedure PtrDec( var P, Delta : Word)
12366:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12367:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
      SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12368:  Function ovMinI( X, Y : Integer) : Integer
12369:  Function ovMaxI( X, Y : Integer) : Integer
12370:  Function ovMinL( X, Y : LongInt) : LongInt
12371:  Function ovMaxL( X, Y : LongInt) : LongInt
12372:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12373:  Function PartialCompare( const S1, S2 : string) : Boolean
12374:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12375:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12376:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12377:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
      TransparentColor : TColor)
12378:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
      TransparentColor : TColorRef)
12379:  Function ovWidthOf( const R : TRect) : Integer
12380:  Function ovHeightOf( const R : TRect) : Integer
12381:  Procedure ovDebugOutput( const S : string)
12382:  Function GetArrowWidth( Width, Height : Integer) : Integer
12383:  Procedure StripCharSeq( CharSeq : string; var Str : string)
```

```
12384:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12385:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12386:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12387:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12388:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12389:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12390:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12391:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12392:  Function CreateMetaFile( p1 : PChar) : HDC
12393:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12394:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12395:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12396:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12397:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12398:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12399:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12400:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12401:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12402:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12403:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12404:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12405:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
       SrcHeight:Int;Rop:DWORD):BOOL
12406:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12407:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
       SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12408:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12409:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12410:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12411:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12412:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12413:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12414:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12415:  Function UpdateColors( DC : HDC) : BOOL
12416:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12417:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12418:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12419:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12420:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12421:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12422:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
       HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12423:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12424:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12425:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12426:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12427:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12428:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12429:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12430:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12431:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12432:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12433:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12434:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12435:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12436: end;
12437:
12438: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12439: begin
12440:   SIRegister_TOvcAbstractStore(CL);
12441:   //PExPropInfo', '^TExPropInfo // will not work
12442: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12443:   SIRegister_TOvcPropertyList(CL);
12444:   SIRegister_TOvcDataFiler(CL);
12445:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12446:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12447:  Function CreateStoredItem( const CompName, PropName : string) : string
12448:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12449:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12450: end;
12451:
12452: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12453: begin
12454:  'ovsetsize','LongInt'( 16);
12455:  'etSyntax','LongInt'( 0);
12456:  'etSymantic','LongInt'( 1);
12457:  'chCR','Char #13);
12458:  'chLF','Char #10);
12459:  'chLineSeparator',' chCR);
12460:   SIRegister_TCocoError(CL);
12461:   SIRegister_TCommentItem(CL);
12462:   SIRegister_TCommentList(CL);
12463:   SIRegister_TSymbolPosition(CL);
12464: TGenListType', '( glNever, glAlways, glOnError )
12465: TovBitSet', 'set of Integer
12466:   //PStartTable', '^TStartTable // will not work
12467: 'TovCharSet', 'set of AnsiChar
12468: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
```

```
12469: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12470: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12471: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12472: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12473:    +'osition; const Data : string; ErrorType : integer)
12474: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12475: TGetCH', 'Function ( pos : longint) : char
12476: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12477:   SIRegister_TCocoRScanner(CL);
12478:   SIRegister_TCocoRGrammar(CL);
12479:  '_EF','Char #0);
12480:  '_TAB','Char').SetString( #09);
12481:  '_CR','Char').SetString( #13);
12482:  '_LF','Char').SetString( #10);
12483:  '_EL','').SetString( _CR);
12484:  '_EOF','Char').SetString( #26);
12485:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12486:  'minErrDist','LongInt'( 2);
12487:  Function ovPadL( S : string; ch : char; L : integer) : string
12488: end;
12489:
12490:   TFormatSettings = record
12491:     CurrencyFormat: Byte;
12492:     NegCurrFormat: Byte;
12493:     ThousandSeparator: Char;
12494:     DecimalSeparator: Char;
12495:     CurrencyDecimals: Byte;
12496:     DateSeparator: Char;
12497:     TimeSeparator: Char;
12498:     ListSeparator: Char;
12499:     CurrencyString: string;
12500:     ShortDateFormat: string;
12501:     LongDateFormat: string;
12502:     TimeAMString: string;
12503:     TimePMString: string;
12504:     ShortTimeFormat: string;
12505:     LongTimeFormat: string;
12506:     ShortMonthNames: array[1..12] of string;
12507:     LongMonthNames: array[1..12] of string;
12508:     ShortDayNames: array[1..7] of string;
12509:     LongDayNames: array[1..7] of string;
12510:     TwoDigitYearCenturyWindow: Word;
12511:   end;
12512:
12513: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12514: begin
12515:  Function ovFormatSettings : TFormatSettings
12516: end;
12517:
12518: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12519: begin
12520:   TOvcCharSet', 'set of Char
12521:   ovBTable', 'array[0..255] of Byte
12522:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
       HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12523:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12524:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12525:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12526:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12527:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12528:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12529:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12530:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12531:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12532:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12533:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12534:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12535:  Function LoCaseChar( C : Char) : Char
12536:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12537:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12538:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12539:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12540:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12541:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12542:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12543:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12544:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12545:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12546:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12547:  Procedure TrimAllSpacesPChar( P : PChar)
12548:  Function TrimEmbeddedZeros( const S : string) : string
12549:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12550:  Function TrimTrailPrimPChar( S : PChar) : PChar
12551:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12552:  Function TrimTrailingZeros( const S : string) : string
12553:  Procedure TrimTrailingZerosPChar( P : PChar)
12554:  Function UpCaseChar( C : Char) : Char
12555:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12556:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
```

```
12557:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12558: end;
12559:
12560: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12561: begin
12562:   //PRaiseFrame', '^TRaiseFrame // will not work
12563:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12564:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12565:  Procedure SafeCloseHandle( var Handle : THandle)
12566:  Procedure ExchangeInteger( X1, X2 : Integer)
12567:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12568:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12569:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12570:
12571: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12572:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12573: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12574:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12575:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12576:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12577:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12578:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12579:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12580:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12581:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12582:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12583:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12584:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12585:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12586:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12587:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12588:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12589:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12590:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12591:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12592:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12593:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12594:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12595:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12596:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12597:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12598:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12599:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12600:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12601:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12602:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12603:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12604:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12605:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12606:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12607:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12608:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12609:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12610:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12611:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12612:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12613:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12614:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12615:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12616:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12617:       var lpLuid: TLargeInteger): BOOL; stdcall;
12618:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12619:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12620:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12621:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12622:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12623:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12624:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12625:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12626:       var GenerateOnClose: BOOL): BOOL; stdcall;
12627:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12628:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12629:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12630:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12631:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12632:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12633:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12634:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12635:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12636:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12637:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12638:       var phkResult: HKEY): Longint; stdcall;
12639:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12640:       var phkResult: HKEY): Longint; stdcall;
12641:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12642:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12643:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12644:       lpdwDisposition: PDWORD): Longint; stdcall;
12645:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
```

```
12646:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12647:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12648:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12649:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12650:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12651:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12652:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12653:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12654:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12655:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12656:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12657:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12658:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12659:       lpcbClass: PDWORD; lpReserved: Pointer;
12660:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12661:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12662:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12663:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12664:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12665:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12666:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12667:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12668:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12669:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12670:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12671:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12672:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12673:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12674:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12675:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12676:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12677:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12678:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12679:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12680:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12681:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12682:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12683:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12684:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12685:
12686:  Function wAddAtom( lpString : PKOLChar) : ATOM
12687:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12688:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12689:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12690:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
      lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12691:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12692:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12693:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12694:  Function wCreateDirectoryEx(lpTemplateDirectory,
      lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12695:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
      bInitialState:BOOL;lpName:PKOLChar):THandle;
12696:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
      PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12697:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
      dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12698:  Function wCreateHardLink(lpFileName,
      lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12699:  Function
      CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12700:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
      nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12701:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
      lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
      Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
      lpProcessInfo:TProcessInformation):BOOL
12702:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
      Longint; lpName : PKOLChar) : THandle
12703:  Function
      wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12704:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12705:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12706:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12707:  //Function
      wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12708:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12709:  //Function
      wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12710:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12711:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12712:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12713:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12714:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12715:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12716:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
      dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
```

```
12717:  Function wFindAtom( lpString : PKOLChar) : ATOM
12718:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12719:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12720:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12721:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12722:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12723:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12724:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12725:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12726:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12727:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12728:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12729:  Function wGetCommandLine : PKOLChar
12730:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12731:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12732:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12733:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12734:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12735:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12736:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12737:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12738:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12739:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12740:  Function wGetEnvironmentStrings : PKOLChar
12741:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12742:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12743:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12744:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12745:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12746:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12747:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12748:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12749:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12750:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12751:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12752:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12753:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12754:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
        nSize:DWORD; lpFileName : PKOLChar) : DWORD
12755:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12756:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12757:  Function wGetProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12758:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12759:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12760: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
        lpCharType):BOOL
12761:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12762:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12763:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12764: //Function
        wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12765: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12766: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
        : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
        lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12767:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12768:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12769:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12770:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12771:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12772:  Function
        wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12773:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12774:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12775:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12776:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12777:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
        TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12778:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12779:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12780:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12781:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12782:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12783:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12784:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
        lpNumberOfEventsRead:DWORD):BOOL;
```

```
12785:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12786:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12787:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
        lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12788:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
        lpNumbOfEventsRead:DWORD):BOOL;
12789:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
        : TCoord; var lpReadRegion : TSmallRect) : BOOL
12790:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12791:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12792:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
        lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12793:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12794:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12795:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12796:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12797:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12798:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12799:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12800:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12801:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12802:  //Function wUpdateResource(hUpdate:THandle;lpType,
        lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12803:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12804:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12805:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
        DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12806:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
        var lpNumberOfEventsWritten : DWORD) : BOOL
12807:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
        TCoord; var lpWriteRegion : TSmallRect) : BOOL
12808:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12809:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12810:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12811:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12812:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12813:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12814:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12815:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12816:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12817:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12818:  Function wlstrlen( lpString : PKOLChar) : Integer
12819:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
        PNetConnectInfoStruct) : DWORD
12820:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12821:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12822:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12823:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12824:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12825:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12826:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12827:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12828:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12829:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
        : PKOLChar; nNameBufSize : DWORD) : DWORD
12830:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12831:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12832:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12833: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
12834:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12835: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12836: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12837:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12838:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12839:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12840:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12841:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12842: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12843: //Func wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12844: //Func wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12845:  Function wAddFontResource( FileName : PKOLChar) : Integer
12846:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12847:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12848:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12849:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12850:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12851: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
```

```
12852:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
        fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
        fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12853:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12854:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12855: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12856:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12857:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12858: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
        pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12859: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12860:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12861:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12862:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12863:  //Function wExtTextOut(DC:HDC;X,
        Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12864:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12865:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12866:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12867:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12868: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12869: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12870:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12871:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12872: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12873: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
        lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12874:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12875: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12876:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12877: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12878:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12879:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12880:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12881:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12882:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12883:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12884:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12885:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12886:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12887:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12888:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12889:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12890:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12891:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12892:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12893:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12894:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12895:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12896:  //Function
        wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12897:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12898: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
        dwFlags : DWORD; lParam : Pointer) : Longint
12899:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12900:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12901:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12902:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12903:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12904:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12905: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12906:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12907:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12908:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12909:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12910:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12911:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12912:  //Function wCreateDesktop(lpszDesktop,
        lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12913:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
        HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12914:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
        lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12915:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
        hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12916: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
        nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12917:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
        dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12918:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12919:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12920:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12921:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12922:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
        : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12923:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
        : TFNDlgProc; dwInitParam : LPARAM) : Integer
12924:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12925:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
        nIDStaticPath:Integer;uFileType:UINT):Integer;
```

```
12926:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
       nIDStaticPath:Int;uFiletype:UINT):Int;
12927:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12928:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12929: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
       cy:Int;Flags:UINT):BOOL;
12930:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12931:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12932:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12933:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
       pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12934: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12935:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12936:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12937:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12938:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12939:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12940:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12941:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12942:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12943:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12944:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12945:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12946:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
       lpnTabStopPositions) : DWORD
12947: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
       lpnLengthNeed:DWORD)BOOL;
12948:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12949:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12950:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12951:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12952: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
       nHeigt:Int):BOOL;
12953:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12954:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12955:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12956:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12957:  Function wIsCharLower( ch : KOLChar) : BOOL
12958:  Function wIsCharUpper( ch : KOLChar) : BOOL
12959:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12960:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12961:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12962:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12963:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12964:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12965:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12966:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12967:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12968:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12969:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12970:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12971:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
12972:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
12973:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12974:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
12975:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12976:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
12977:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
12978:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12979:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12980:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12981:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
12982:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
12983:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12984:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12985:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
12986: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
12987: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
12988:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
12989: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12990:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
12991:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
12992:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12993:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12994:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
       lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
12995:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
       lpdwResult:DWORD): LRESULT
12996:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12997:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
12998:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
12999:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
13000:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
13001: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
13002:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
13003:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
13004:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
13005:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13006: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
```

```
13007:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
       lpnTabStopPositions,nTabOrigin:Int):Longint;
13008:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
13009:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
13010:  Function wVkKeyScan( ch : KOLChar) : SHORT
13011:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13012:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
13013:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
13014:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
13015:
13016: //TestDrive!
13017: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13018:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13019:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
13020:  Function GetLocalUserSidStr( const UserName : string) : string
13021:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
13022:  Function Impersonate2User( const domain : string; const user : string) : boolean
13023:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
13024:  Function KillProcessbyname( const exename : string; var found : integer) : integer
13025:  Function getWinProcessList : TStringList
13026:  function WaitTilClose(hWnd: Integer): Integer;
13027:  function DoUserMsgs: Boolean;
13028:  function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13029:  procedure ShowMsg(hParent: Integer; const Mess, Title: String);  //modal but NOT blockable
13030:  procedure DeleteMsgForm(Handle: Integer);
13031:  procedure DisableForms;
13032:  function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13033: end;
13034:
13035: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13036: begin
13037:  'AfMaxSyncSlots','LongInt'( 64);
13038:  'AfSynchronizeTimeout','LongInt'( 2000);
13039:   TAfSyncSlotID', 'DWORD
13040:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13041:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
13042:   TAfSafeDirectSyncEvent', 'Procedure
13043:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13044:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
13045:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
13046:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
13047:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
13048:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
13049:  Function AfIsSyncMethod : Boolean
13050:  Function AfSyncWnd : HWnd
13051:  Function AfSyncStatistics : TAfSyncStatistics
13052:  Procedure AfClearSyncStatistics
13053: end;
13054:
13055: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13056: begin
13057:  'fBinary','LongWord')( $00000001);
13058:  'fParity','LongWord'( $00000002);
13059:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13060:  'fOutxDsrFlow','LongWord')( $00000008);
13061:  'fDtrControl','LongWord')( $00000030);
13062:  'fDtrControlDisable','LongWord')( $00000000);
13063:  'fDtrControlEnable','LongWord')( $00000010);
13064:  'fDtrControlHandshake','LongWord')( $00000020);
13065:  'fDsrSensitivity','LongWord')( $00000040);
13066:  'fTXContinueOnXoff','LongWord')( $00000080);
13067:  'fOutX','LongWord')( $00000100);
13068:  'fInX','LongWord')( $00000200);
13069:  'fErrorChar','LongWord')( $00000400);
13070:  'fNull','LongWord')( $00000800);
13071:  'fRtsControl','LongWord')( $00003000);
13072:  'fRtsControlDisable','LongWord')( $00000000);
13073:  'fRtsControlEnable','LongWord')( $00001000);
13074:  'fRtsControlHandshake','LongWord')( $00002000);
13075:  'fRtsControlToggle','LongWord')( $00003000);
13076:  'fAbortOnError','LongWord')( $00004000);
13077:  'fDummy2','LongWord')( $FFFF8000);
13078:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13079:   FindClass('TOBJECT'),'EAfComPortCoreError
13080:   FindClass('TOBJECT'),'TAfComPortCore
13081:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13082:    +'tKind : TAfCoreEvent; Data : DWORD)
13083:   SIRegister_TAfComPortCoreThread(CL);
13084:   SIRegister_TAfComPortEventThread(CL);
13085:   SIRegister_TAfComPortWriteThread(CL);
13086:   SIRegister_TAfComPortCore(CL);
13087:  Function FormatDeviceName( PortNumber : Integer) : string
13088: end;
13089:
13090: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13091: begin
13092:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13093:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13094:   SIRegister_TApplicationFileIO(CL);
```

```
13095:   TDataFileCapability', '( dfcRead, dfcWrite )
13096:   TDataFileCapabilities', 'set of TDataFileCapability
13097:   SIRegister_TDataFile(CL);
13098:   //TDataFileClass', 'class of TDataFile
13099:  Function ApplicationFileIODefined : Boolean
13100:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13101:  Function FileStreamExists(const fileName: String) : Boolean
13102:  //Procedure Register
13103: end;
13104:
13105: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13106: begin
13107:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13108:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13109:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13110:   TALFBXScale', 'Integer
13111:   FindClass('TOBJECT'),'EALFBXConvertError
13112:   SIRegister_EALFBXError(CL);
13113:   SIRegister_EALFBXException(CL);
13114:   FindClass('TOBJECT'),'EALFBXGFixError
13115:   FindClass('TOBJECT'),'EALFBXDSQLError
13116:   FindClass('TOBJECT'),'EALFBXDynError
13117:   FindClass('TOBJECT'),'EALFBXGBakError
13118:   FindClass('TOBJECT'),'EALFBXGSecError
13119:   FindClass('TOBJECT'),'EALFBXLicenseError
13120:   FindClass('TOBJECT'),'EALFBXGStatError
13121:   //EALFBXExceptionClass', 'class of EALFBXError
13122:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13123:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13124:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13125:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13126:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13127:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13128:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13129:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13130:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13131:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13132:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13133:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13134:   TALFBXTransParams', 'set of TALFBXTransParam
13135:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13136:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13137:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13138: 'cALFBXMaxParamLength','LongInt'( 125);
13139:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13140:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13141:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13142:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13143:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13144:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13145:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13146:   SIRegister_TALFBXSQLDA(CL);
13147:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13148:   SIRegister_TALFBXPoolStream(CL);
13149:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13150:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13151:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13152:   //TALFBXArrayDesc', 'TISCArrayDesc
13153:   //TALFBXBlobDesc', 'TISCBlobDesc
13154:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13155:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13156:   SIRegister_TALFBXSQLResult(CL);
13157:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13158:   SIRegister_TALFBXSQLParams(CL);
13159:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13160:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13161:    +'atementType : TALFBXStatementType; end
13162:   FindClass('TOBJECT'),'TALFBXLibrary
13163:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13164:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13165:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13166:    //+' Excep : EALFBXExceptionClass)
13167:   SIRegister_TALFBXLibrary(CL);
13168:  'cAlFBXDateOffset','LongInt'( 15018);
13169:  'cALFBXTimeCoeff','LongInt'( 864000000);
13170:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13171:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13172:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13173:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13174:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13175:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13176:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13177:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13178:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13179:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13180:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13181:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13182:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13183:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
```

```
13184: end;
13185:
13186: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13187: begin
13188:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13189:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13190:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13191:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13192:    +'teger; First : Integer; CacheThreshold : Integer; end
13193:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13194:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13195:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13196:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13197:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13198:   SIRegister_TALFBXClient(CL);
13199:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13200:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13201:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13202:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13203:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13204:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13205:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13206:   SIRegister_TALFBXConnectionPoolClient(CL);
13207:   SIRegister_TALFBXEventThread(CL);
13208:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13209: end;
13210:
13211: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13212: begin
13213: _OSVERSIONINFOA = record
13214:     dwOSVersionInfoSize: DWORD;
13215:     dwMajorVersion: DWORD;
13216:     dwMinorVersion: DWORD;
13217:     dwBuildNumber: DWORD;
13218:     dwPlatformId: DWORD;
13219:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13220:   end;
13221:  TOSVersionInfoA', '_OSVERSIONINFOA
13222:   TOSVersionInfo', 'TOSVersionInfoA
13223:  'WS_EX_RIGHT','LongWord')( $00001000);
13224:  'WS_EX_LEFT','LongWord')( $00000000);
13225:  'WS_EX_RTLREADING','LongWord')( $00002000);
13226:  'WS_EX_LTRREADING','LongWord')( $00000000);
13227:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13228:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13229:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13230:  'LAYOUT_RTL','LongWord')( $00000001);
13231:  'LAYOUT_BTT','LongWord')( $00000002);
13232:  'LAYOUT_VBH','LongWord')( $00000004);
13233:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13234:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13235:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13236:  Function GetLayout( dc : hdc) : DWORD
13237:  Function IsBidi : Boolean
13238:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13239:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13240:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13241:  Function GetPriorityClass( hProcess : THandle) : DWORD
13242:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13243:  Function CloseClipboard : BOOL
13244:  Function GetClipboardSequenceNumber : DWORD
13245:  Function GetClipboardOwner : HWND
13246:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13247:  Function GetClipboardViewer : HWND
13248:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13249:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13250:  Function GetClipboardData( uFormat : UINT) : THandle
13251:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13252:  Function CountClipboardFormats : Integer
13253:  Function EnumClipboardFormats( format : UINT) : UINT
13254:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13255:  Function EmptyClipboard : BOOL
13256:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13257:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13258:  Function GetOpenClipboardWindow : HWND
13259:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13260:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13261:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13262:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13263:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13264:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13265:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13266:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13267:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13268: end;
13269:
13270: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13271: begin
13272:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
```

```
13273:  Function GetTemporaryFilesPath : String
13274:  Function GetTemporaryFileName : String
13275:  Function FindFileInPaths( const fileName, paths : String) : String
13276:  Function PathsToString( const paths : TStrings) : String
13277:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13278:  //Function MacroExpandPath( const aPath : String) : String
13279: end;
13280:
13281: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13282: begin
13283:   SIRegister_TALMultiPartBaseContent(CL);
13284:   SIRegister_TALMultiPartBaseContents(CL);
13285:   SIRegister_TAlMultiPartBaseStream(CL);
13286:   SIRegister_TALMultipartBaseEncoder(CL);
13287:   SIRegister_TALMultipartBaseDecoder(CL);
13288:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13289:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13290:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13291: end;
13292:
13293: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13294: begin
13295:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13296:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13297:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13298:  Function aAllocPadedMem( Size : Cardinal) : TObject
13299:  Procedure aFreePadedMem( var P : TObject);
13300:  Procedure aFreePadedMem1( var P : PChar);
13301:  Function aCheckPadedMem( P : Pointer) : Byte
13302:  Function aGetPadMemSize( P : Pointer) : Cardinal
13303:  Function aAllocMem( Size : Cardinal) : Pointer
13304:  Function aStrLen( const Str : PChar) : Cardinal
13305:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13306:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13307:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13308:  Function aStrEnd( const Str : PChar) : PChar
13309:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13310:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13311:  Function aPCharLength( const Str : PChar) : Cardinal
13312:  Function aPCharUpper( Str : PChar) : PChar
13313:  Function aPCharLower( Str : PChar) : PChar
13314:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13315:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13316:  Function aCopyTail( const S : String; Len : Integer) : String
13317:  Function aInt2Thos( I : Int64) : String
13318:  Function aUpperCase( const S : String) : String
13319:  Function aLowerCase( const S : string) : String
13320:  Function aCompareText( const S1, S2 : string) : Integer
13321:  Function aSameText( const S1, S2 : string) : Boolean
13322:  Function aInt2Str( Value : Int64) : String
13323:  Function aStr2Int( const Value : String) : Int64
13324:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13325:  Function aGetFileExt( const FileName : String) : String
13326:  Function aGetFilePath( const FileName : String) : String
13327:  Function aGetFileName( const FileName : String) : String
13328:  Function aChangeExt( const FileName, Extension : String) : String
13329:  Function aAdjustLineBreaks( const S : string) : string
13330:  Function aGetWindowStr( WinHandle : HWND) : String
13331:  Function aDiskSpace( Drive : String) : TdriveSize
13332:  Function aFileExists( FileName : String) : Boolean
13333:  Function aFileSize( FileName : String) : Int64
13334:  Function aDirectoryExists( const Name : string) : Boolean
13335:  Function aSysErrorMessage( ErrorCode : Integer) : string
13336:  Function aShortPathName( const LongName : string) : string
13337:  Function aGetWindowVer : TWinVerRec
13338:  procedure InitDriveSpacePtr;
13339: end;
13340:
13341: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13342: begin
13343:  aZero','LongInt'( 0);
13344:  'makeappDEF','LongInt'( - 1);
13345:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13346:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13347:  'CS_KEYCVTWINDOW','LongInt'( 4);
13348:  'CS_DBLCLKS','LongInt'( 8);
13349:  'CS_OWNDC','LongWord')( $20);
13350:  'CS_CLASSDC','LongWord')( $40);
13351:  'CS_PARENTDC','LongWord')( $80);
13352:  'CS_NOKEYCVT','LongWord')( $100);
13353:  'CS_NOCLOSE','LongWord')( $200);
13354:  'CS_SAVEBITS','LongWord')( $800);
13355:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13356:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13357:  'CS_GLOBALCLASS','LongWord')( $4000);
13358:  'CS_IME','LongWord')( $10000);
13359:  'CS_DROPSHADOW','LongWord')( $20000);
13360:   //PPanelFunc', '^TPanelFunc // will not work
13361:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
```

```
13362:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13363:   TFontLooks', 'set of TFontLook
13364:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13365:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13366:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13367:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13368:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13369:  Procedure RunMsgLoop( Show : Boolean)
13370:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13371:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
      ID_Number:Cardinal;hFont:Int):Int;
13372:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13373:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13374:  Function MakePanel(Left,Top,Width,Height,
      hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13375:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13376:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13377:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13378: end;
13379:
13380: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13381: begin
13382:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13383:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13384:   TScreenSaverOptions', 'set of TScreenSaverOption
13385:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
      ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13386:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13387:   SIRegister_TScreenSaver(CL);
13388:  //Procedure Register
13389:  Procedure SetScreenSaverPassword
13390: end;
13391:
13392: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13393: begin
13394:   FindClass('TOBJECT'),'TXCollection
13395:   SIRegister_EFilerException(CL);
13396:   SIRegister_TXCollectionItem(CL);
13397:   //TXCollectionItemClass', 'class of TXCollectionItem
13398:   SIRegister_TXCollection(CL);
13399:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13400:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13401:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13402:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13403:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13404:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13405: end;
13406:
13407: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13408: begin
13409:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13410:  Procedure xglMapTexCoordToNull
13411:  Procedure xglMapTexCoordToMain
13412:  Procedure xglMapTexCoordToSecond
13413:  Procedure xglMapTexCoordToDual
13414:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13415:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13416:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13417:  Procedure xglBeginUpdate
13418:  Procedure xglEndUpdate
13419:  Procedure xglPushState
13420:  Procedure xglPopState
13421:  Procedure xglForbidSecondTextureUnit
13422:  Procedure xglAllowSecondTextureUnit
13423:  Function xglGetBitWiseMapping : Cardinal
13424: end;
13425:
13426: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13427: begin
13428:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13429:   TBaseListOptions', 'set of TBaseListOption
13430:   SIRegister_TBaseList(CL);
13431:   SIRegister_TBaseVectorList(CL);
13432:   SIRegister_TAffineVectorList(CL);
13433:   SIRegister_TVectorList(CL);
13434:   SIRegister_TTexPointList(CL);
13435:   SIRegister_TXIntegerList(CL);
13436:   //PSingleArrayList', '^TSingleArrayList // will not work
13437:   SIRegister_TSingleList(CL);
13438:   SIRegister_TByteList(CL);
13439:   SIRegister_TQuaternionList(CL);
13440:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13441:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13442:  Procedure FastQuickSortLists(startIndex,
      endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13443: end;
13444:
13445: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13446: begin
```

```
13447:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13448:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13449:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
       indices:TIntegerList;list:TAffineVectorList);
13450:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
       indices:TIntegerList;list:TAffineVectorList);
13451:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
       normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13452:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13453:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13454:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13455:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13456:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13457:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13458:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13459:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13460:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13461:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
       edgesTriangles : TIntegerList) : TIntegerList
13462:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13463:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
       TPersistentObjectList;
13464:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13465:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
       TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13466: end;
13467:
13468: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13469: begin
13470:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13471:  Procedure FreeMemAndNil( var P : TObject)
13472:  Function PCharOrNil( const S : string) : PChar
13473:   SIRegister_TJclReferenceMemoryStream(CL);
13474:   FindClass('TOBJECT'),'EJclVMTError
13475:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13476:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13477:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13478:   PDynamicIndexList', '^TDynamicIndexList // will not work
13479:   PDynamicAddressList', '^TDynamicAddressList // will not work
13480:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13481:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13482:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13483:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13484:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13485:  Function GetInitTable( AClass : TClass) : PTypeInfo
13486:   PFieldEntry', '^TFieldEntry // will not work}
13487:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13488:  Function JIsClass( Address : Pointer) : Boolean
13489:  Function JIsObject( Address : Pointer) : Boolean
13490:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13491:   TDigitCount', 'Integer
13492:   SIRegister_TJclNumericFormat(CL);
13493:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13494:   TTextHandler', 'Procedure ( const Text : string)
13495: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13496:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13497:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13498:  Function ReadKey : Char   //to and from the DOS console !
13499:   TModuleHandle', 'HINST
13500:   //TModuleHandle', 'Pointer
13501:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13502:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13503:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13504:  Procedure UnloadModule( var Module : TModuleHandle)
13505:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13506:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13507:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13508:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13509:   FindClass('TOBJECT'),'EJclConversionError
13510:  Function JStrToBoolean( const S : string) : Boolean
13511:  Function JBooleanToStr( B : Boolean) : string
13512:  Function JIntToBool( I : Integer) : Boolean
13513:  Function JBoolToInt( B : Boolean) : Integer
13514:  'ListSeparator','String ';
13515:  'ListSeparator1','String ':
13516:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13517:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13518:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13519:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13520:  Function ListItemCount( const List, Separator : string) : Integer
13521:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13522:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13523:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13524:  Function SystemTObjectInstance : LongWord
13525:  Function IsCompiledWithPackages : Boolean
13526:  Function JJclGUIDToString( const GUID : TGUID) : string
13527:  Function JJclStringToGUID( const S : string) : TGUID
13528:   SIRegister_TJclIntfCriticalSection(CL);
```

```
13529:   SIRegister_TJclSimpleLog(CL);
13530:  Procedure InitSimpleLog( const ALogFileName : string)
13531: end;
13532:
13533: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13534: begin
13535:   FindClass('TOBJECT'),'EJclBorRADException
13536:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13537:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13538:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13539:   TJclBorRADToolPath', 'string
13540:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13541:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13542:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13543:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13544:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13545:  BorRADToolRepositoryFormsPage','String 'Forms
13546:  BorRADToolRepositoryProjectsPage','String 'Projects
13547:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13548:  BorRADToolRepositoryObjectType','String 'Type
13549:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13550:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13551:  BorRADToolRepositoryObjectName','String 'Name
13552:  BorRADToolRepositoryObjectPage','String 'Page
13553:  BorRADToolRepositoryObjectIcon','String 'Icon
13554:  BorRADToolRepositoryObjectDescr','String 'Description
13555:  BorRADToolRepositoryObjectAuthor','String 'Author
13556:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13557:  BorRADToolRepositoryObjectDesigner','String 'Designer
13558:  BorRADToolRepositoryDesignerDfm','String 'dfm
13559:  BorRADToolRepositoryDesignerXfm','String 'xfm
13560:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13561:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13562:  SourceExtensionDelphiPackage','String '.dpk
13563:  SourceExtensionBCBPackage','String '.bpk
13564:  SourceExtensionDelphiProject','String '.dpr
13565:  SourceExtensionBCBProject','String '.bpr
13566:  SourceExtensionBDSProject','String '.bdsproj
13567:  SourceExtensionDProject','String '.dproj
13568:  BinaryExtensionPackage','String '.bpl
13569:  BinaryExtensionLibrary','String '.dll
13570:  BinaryExtensionExecutable','String '.exe
13571:  CompilerExtensionDCP','String '.dcp
13572:  CompilerExtensionBPI','String '.bpi
13573:  CompilerExtensionLIB','String '.lib
13574:  CompilerExtensionTDS','String '.tds
13575:  CompilerExtensionMAP','String '.map
13576:  CompilerExtensionDRC','String '.drc
13577:  CompilerExtensionDEF','String '.def
13578:  SourceExtensionCPP','String '.cpp
13579:  SourceExtensionH','String '.h
13580:  SourceExtensionPAS','String '.pas
13581:  SourceExtensionDFM','String '.dfm
13582:  SourceExtensionXFM','String '.xfm
13583:  SourceDescriptionPAS','String 'Pascal source file
13584:  SourceDescriptionCPP','String 'C++ source file
13585:  DesignerVCL','String 'VCL
13586:  DesignerCLX','String 'CLX
13587:  ProjectTypePackage','String 'package
13588:  ProjectTypeLibrary','String 'library
13589:  ProjectTypeProgram','String 'program
13590:  Personality32Bit','String '32 bit
13591:  Personality64Bit','String '64 bit
13592:  PersonalityDelphi','String 'Delphi
13593:  PersonalityDelphiDotNet','String 'Delphi.net
13594:  PersonalityBCB','String 'C++Builder
13595:  PersonalityCSB','String 'C#Builder
13596:  PersonalityVB','String 'Visual Basic
13597:  PersonalityDesign','String 'Design
13598:  PersonalityUnknown','String 'Unknown personality
13599:  PersonalityBDS','String 'Borland Developer Studio
13600:  DOFDirectoriesSection','String 'Directories
13601:  DOFUnitOutputDirKey','String 'UnitOutputDir
13602:  DOFSearchPathName','String 'SearchPath
13603:  DOFConditionals','String 'Conditionals
13604:  DOFLinkerSection','String 'Linker
13605:  DOFPackagesKey','String 'Packages
13606:  DOFCompilerSection','String 'Compiler
13607:  DOFPackageNoLinkKey','String 'PackageNoLink
13608:  DOFAdditionalSection','String 'Additional
13609:  DOFOptionsKey','String 'Options
13610:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13611:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13612:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13613:   TJclBorPersonalities', 'set of TJclBorPersonality
13614:   TJclBorDesigner', '( bdVCL, bdCLX )
13615:   TJclBorDesigners', 'set of TJClBorDesigner
13616:   TJclBorPlatform', '( bp32bit, bp64bit )
13617:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
```

```
13618:   SIRegister_TJclBorRADToolInstallationObject(CL);
13619:   SIRegister_TJclBorlandOpenHelp(CL);
13620:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13621:   TJclHelp2Objects', 'set of TJclHelp2Object
13622:   SIRegister_TJclHelp2Manager(CL);
13623:   SIRegister_TJclBorRADToolIdeTool(CL);
13624:   SIRegister_TJclBorRADToolIdePackages(CL);
13625:   SIRegister_IJclCommandLineTool(CL);
13626:   FindClass('TOBJECT'),'EJclCommandLineToolError
13627:   SIRegister_TJclCommandLineTool(CL);
13628:   SIRegister_TJclBorlandCommandLineTool(CL);
13629:   SIRegister_TJclBCC32(CL);
13630:   SIRegister_TJclDCC32(CL);
13631:   TJclDCC', 'TJclDCC32
13632:   SIRegister_TJclBpr2Mak(CL);
13633:   SIRegister_TJclBorlandMake(CL);
13634:   SIRegister_TJclBorRADToolPalette(CL);
13635:   SIRegister_TJclBorRADToolRepository(CL);
13636:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13637:   TCommandLineTools', 'set of TCommandLineTool
13638:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13639:   SIRegister_TJclBorRADToolInstallation(CL);
13640:   SIRegister_TJclBCBInstallation(CL);
13641:   SIRegister_TJclDelphiInstallation(CL);
13642:   SIRegister_TJclDCCIL(CL);
13643:   SIRegister_TJclBDSInstallation(CL);
13644:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13645:   SIRegister_TJclBorRADToolInstallations(CL);
13646:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13647:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13648:  Function IsDelphiPackage( const FileName : string) : Boolean
13649:  Function IsDelphiProject( const FileName : string) : Boolean
13650:  Function IsBCBPackage( const FileName : string) : Boolean
13651:  Function IsBCBProject( const FileName : string) : Boolean
13652:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13653:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13654:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
      Descript:PString;
13655:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
      Descript:PString
13656:  function SamePath(const Path1, Path2: string): Boolean;
13657: end;
13658:
13659: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13660: begin
13661:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13662:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13663:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13664:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13665:  'LPathSeparator','String '/
13666:  'LDirDelimiter','String '/
13667:  'LDirSeparator','String ':
13668:  'JXPathDevicePrefix','String '\\.\
13669:  'JXPathSeparator','String '\
13670:  'JXDirDelimiter','String '\
13671:  'JXDirSeparator','String ';
13672:  'JXPathUncPrefix','String '\\
13673:  'faNormalFile','LongWord')( $00000080);
13674:  //'faUnixSpecific',' faSymLink);
13675:   JXTCompactPath', '( cpCenter, cpEnd )
13676:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13677:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13678:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13679:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13680:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13681:
13682:  Function jxPathAddSeparator( const Path : string) : string
13683:  Function jxPathAddExtension( const Path, Extension : string) : string
13684:  Function jxPathAppend( const Path, Append : string) : string
13685:  Function jxPathBuildRoot( const Drive : Byte) : string
13686:  Function jxPathCanonicalize( const Path : string) : string
13687:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13688:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13689:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13690:  Function jxPathExtractFileDirFixed( const S : string) : string
13691:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13692:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13693:  Function jxPathGetDepth( const Path : string) : Integer
13694:  Function jxPathGetLongName( const Path : string) : string
13695:  Function jxPathGetShortName( const Path : string) : string
13696:  Function jxPathGetLongName( const Path : string) : string
13697:  Function jxPathGetShortName( const Path : string) : string
13698:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13699:  Function jxPathGetTempPath : string
13700:  Function jxPathIsAbsolute( const Path : string) : Boolean
13701:  Function jxPathIsChild( const Path, Base : string) : Boolean
13702:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13703:  Function jxPathIsUNC( const Path : string) : Boolean
13704:  Function jxPathRemoveSeparator( const Path : string) : string
```

```
13705:  Function jxPathRemoveExtension( const Path : string) : string
13706:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13707:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13708:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13709:   JxTFileListOptions', 'set of TFileListOption
13710:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13711:   TFileHandler', 'Procedure ( const FileName : string)
13712:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13713:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13714:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
       AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
       FileMatchFunc:TFileMatchFunc):Bool;
13715:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13716:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13717:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13718:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13719:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
       RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13720:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
       IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13721:  Procedure jxCreateEmptyFile( const FileName : string)
13722:  Function jxCloseVolume( var Volume : THandle) : Boolean
13723:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13724:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13725:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13726:  Function jxDelTree( const Path : string) : Boolean
13727:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13728:  Function jxDiskInDrive( Drive : Char) : Boolean
13729:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13730:  Function jxFileCreateTemp( var Prefix : string) : THandle
13731:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13732:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13733:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13734:  Function jxFileExists( const FileName : string) : Boolean
13735:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13736:  Function jxFileRestore( const FileName : string) : Boolean
13737:  Function jxGetBackupFileName( const FileName : string) : string
13738:  Function jxIsBackupFileName( const FileName : string) : Boolean
13739:  Function jxFileGetDisplayName( const FileName : string) : string
13740:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13741:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13742:  Function jxFileGetSize( const FileName : string) : Int64
13743:  Function jxFileGetTempName( const Prefix : string) : string
13744:  Function jxFileGetTypeName( const FileName : string) : string
13745:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13746:  Function jxForceDirectories( Name : string) : Boolean
13747:  Function jxGetDirectorySize( const Path : string) : Int64
13748:  Function jxGetDriveTypeStr( const Drive : Char) : string
13749:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13750:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13751:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13752:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13753:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13754:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
       ResolveSymLinks:Boolean):Integer
13755:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13756:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13757:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13758:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13759:  Function jxGetFileCreation( const FName : string) : TFileTime;
13760:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13761:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13762:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13763:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13764:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13765:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13766:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13767:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13768:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13769:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13770:  Function jxGetModulePath( const Module : HMODULE) : string
13771:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13772:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13773:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13774:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13775:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13776:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13777:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13778:  Function jxOpenVolume( const Drive : Char) : THandle
13779:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13780:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13781:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13782:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13783:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13784:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13785:  Procedure jxShredFile( const FileName : string; Times : Integer)
13786:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13787:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13788:  Function jxSymbolicLinkTarget( const Name : string) : string
```

```
13789:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13790:   SIRegister_TJclCustomFileAttrMask(CL);
13791:   SIRegister_TJclFileAttributeMask(CL);
13792:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13793:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13794:   TFileSearchOptions', 'set of TFileSearchOption
13795:   TFileSearchTaskID', 'Integer
13796:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13797:    +'hTaskID; const Aborted : Boolean)
13798:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13799:   SIRegister_IJclFileEnumerator(CL);
13800:   SIRegister_TJclFileEnumerator(CL);
13801:  Function JxFileSearch : IJclFileEnumerator
13802:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13803:   JxTFileFlags', 'set of TFileFlag
13804:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13805:   SIRegister_TJclFileVersionInfo(CL);
13806:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13807:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13808:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13809:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13810:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13811:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13812:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13813:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13814:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13815:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13816:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13817:   SIRegister_TJclTempFileStream(CL);
13818:   FindClass('TOBJECT'),'TJclCustomFileMapping
13819:   SIRegister_TJclFileMappingView(CL);
13820:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13821:   SIRegister_TJclCustomFileMapping(CL);
13822:   SIRegister_TJclFileMapping(CL);
13823:   SIRegister_TJclSwapFileMapping(CL);
13824:   SIRegister_TJclFileMappingStream(CL);
13825:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13826:   //PPCharArray', '^TPCharArray // will not work
13827:   SIRegister_TJclMappedTextReader(CL);
13828:   SIRegister_TJclFileMaskComparator(CL);
13829:   FindClass('TOBJECT'),'EJclPathError
13830:   FindClass('TOBJECT'),'EJclFileUtilsError
13831:   FindClass('TOBJECT'),'EJclTempFileStreamError
13832:   FindClass('TOBJECT'),'EJclTempFileStreamError
13833:   FindClass('TOBJECT'),'EJclFileMappingError
13834:   FindClass('TOBJECT'),'EJclFileMappingViewError
13835:  Function jxPathGetLongName2( const Path : string) : string
13836:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13837:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13838:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13839:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13840:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13841:  Procedure jxPathListAddItems( var List : string; const Items : string)
13842:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13843:  Procedure jxPathListDelItems( var List : string; const Items : string)
13844:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13845:  Function jxPathListItemCount( const List : string) : Integer
13846:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13847:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13848:  Function jxPathListItemIndex( const List, Item : string) : Integer
13849:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13850:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13851:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13852:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string) : Integer
13853: end;
13854:
13855: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13856: begin
13857:  'UTF8FileHeader','String #$ef#$bb#$bf);
13858:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13859:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13860:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13861:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13862:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13863:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13864:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13865:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13866:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13867:  Function lFileIsReadable( const AFilename : string) : boolean
13868:  Function lFileIsWritable( const AFilename : string) : boolean
13869:  Function lFileIsText( const AFilename : string) : boolean
13870:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13871:  Function lFileIsExecutable( const AFilename : string) : boolean
13872:  Function lFileIsSymlink( const AFilename : string) : boolean
13873:  Function lFileIsHardLink( const AFilename : string) : boolean
```

```
13874:  Function lFileSize( const Filename : string) : int64;
13875:  Function lGetFileDescription( const AFilename : string) : string
13876:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13877:  Function lTryReadAllLinks( const Filename : string) : string
13878:  Function lDirPathExists( const FileName : String) : Boolean
13879:  Function lForceDirectory( DirectoryName : string) : boolean
13880:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13881:  Function lProgramDirectory : string
13882:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13883:  Function lExtractFileNameOnly( const AFilename : string) : string
13884:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13885:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13886:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13887:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13888:  Function lAppendPathDelim( const Path : string) : string
13889:  Function lChompPathDelim( const Path : string) : string
13890:  Function lTrimFilename( const AFilename : string) : string
13891:  Function lCleanAndExpandFilename( const Filename : string) : string
13892:  Function lCleanAndExpandDirectory( const Filename : string) : string
13893:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13894:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
      AlwaysRequireSharedBaseFolder : Boolean) : string
13895:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13896:  Function lFileIsInPath( const Filename, Path : string) : boolean
13897:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13898:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13899:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13900:  'AllDirectoryEntriesMask','String '*
13901:  Function lGetAllFilesMask : string
13902:  Function lGetExeExt : string
13903:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
      TSearchFileInPathFlags) : string
13904:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
      TSearchFileInPathFlags) : TStrings
13905:  Function lFindDiskFilename( const Filename : string) : string
13906:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13907:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13908:  Function lGetDarwinSystemFilename( Filename : string) : string
13909:   SIRegister_TFileIterator(CL);
13910:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13911:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13912:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13913:   SIRegister_TFileSearcher(CL);
13914:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13915:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13916:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13917:  // TCopyFileFlags', 'set of TCopyFileFlag
13918:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13919:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13920:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13921:  Function lReadFileToString( const Filename : string) : string
13922:  Function lGetTempFilename( const Directory, Prefix : string) : string
13923:  {Function NeedRTLAnsi : boolean
13924:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13925:  Function UTF8ToSys( const s : string) : string
13926:  Function SysToUTF8( const s : string) : string
13927:  Function ConsoleToUTF8( const s : string) : string
13928:  Function UTF8ToConsole( const s : string) : string}
13929:  Function FileExistsUTF8( const Filename : string) : boolean
13930:  Function FileAgeUTF8( const FileName : string) : Longint
13931:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13932:  Function ExpandFileNameUTF8( const FileName : string) : string
13933:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13934:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13935:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13936:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13937:  Procedure FindCloseUTF8( var F : TSearchrec)
13938:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13939:  Function FileGetAttrUTF8( const FileName : String) : Longint
13940:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13941:  Function DeleteFileUTF8( const FileName : String) : Boolean
13942:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13943:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13944:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13945:  Function GetCurrentDirUTF8 : String
13946:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13947:  Function CreateDirUTF8( const NewDir : String) : Boolean
13948:  Function RemoveDirUTF8( const Dir : String) : Boolean
13949:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13950:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13951:  Function FileCreateUTF8( const FileName : string) : THandle;
13952:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13953:  Function ParamStrUTF8( Param : Integer) : string
13954:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13955:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13956:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13957:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13958:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13959: end;
```

```
13960:
13961: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13962: begin
13963:   //VK_F23 = 134;
13964:   //{$EXTERNALSYM VK_F24}
13965:   //VK_F24 = 135;
13966:  TVirtualKeyCode', 'Integer
13967:  'VK_MOUSEWHEELUP','integer'(134);
13968:  'VK_MOUSEWHEELDOWN','integer'(135);
13969:  Function glIsKeyDown( c : Char) : Boolean;
13970:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13971:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13972:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13973:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13974:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13975:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13976: end;
13977:
13978: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13979: begin
13980:   TGLPoint', 'TPoint
13981:   //PGLPoint', '^TGLPoint // will not work
13982:   TGLRect', 'TRect
13983:   //PGLRect', '^TGLRect // will not work
13984:   TDelphiColor', 'TColor
13985:   TGLPicture', 'TPicture
13986:   TGLGraphic', 'TGraphic
13987:   TGLBitmap', 'TBitmap
13988:   //TGraphicClass', 'class of TGraphic
13989:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13990:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
13991:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
13992:    +'Button; Shift : TShiftState; X, Y : Integer)
13993:   TGLMouseMoveEvent', 'TMouseMoveEvent
13994:   TGLKeyEvent', 'TKeyEvent
13995:   TGLKeyPressEvent', 'TKeyPressEvent
13996:   EGLOSError', 'EWin32Error
13997:   EGLOSError', 'EWin32Error
13998:   EGLOSError', 'EOSError
13999:  'glsAllFilter','string'All // sAllFilter
14000:  Function GLPoint( const x, y : Integer) : TGLPoint
14001:  Function GLRGB( const r, g, b : Byte) : TColor
14002:  Function GLColorToRGB( color : TColor) : TColor
14003:  Function GLGetRValue( rgb : DWORD) : Byte
14004:  Function GLGetGValue( rgb : DWORD) : Byte
14005:  Function GLGetBValue( rgb : DWORD) : Byte
14006:  Procedure GLInitWinColors
14007:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
14008:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
14009:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14010:  Procedure GLInformationDlg( const msg : String)
14011:  Function GLQuestionDlg( const msg : String) : Boolean
14012:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
14013:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
14014:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
14015:  Function GLApplicationTerminated : Boolean
14016:  Procedure GLRaiseLastOSError
14017:  Procedure GLFreeAndNil( var anObject: TObject)
14018:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
14019:  Function GLGetCurrentColorDepth : Integer
14020:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14021:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14022:  Procedure GLSleep( length : Cardinal)
14023:  Procedure GLQueryPerformanceCounter( var val : Int64)
14024:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14025:  Function GLStartPrecisionTimer : Int64
14026:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14027:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14028:  Function GLRDTSC : Int64
14029:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14030:  Function GLOKMessageBox( const Text, Caption : string) : Integer
14031:  Procedure GLShowHTMLUrl( Url : String)
14032:  Procedure GLShowCursor( AShow : boolean)
14033:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14034:  Procedure GLGetCursorPos( var point : TGLPoint)
14035:  Function GLGetScreenWidth : integer
14036:  Function GLGetScreenHeight : integer
14037:  Function GLGetTickCount : int64
14038:  function RemoveSpaces(const str : String) : String;
14039:   TNormalMapSpace','( nmsObject, nmsTangent )
14040:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14041:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14042:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
14043:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
14044:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14045: end;
```

```
14046:
14047: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14048: begin
14049:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14050:  // PGLStarRecord', '^TGLStarRecord // will not work
14051:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14052:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14053:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14054: end;
14055:
14056:
14057: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14058: begin
14059:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14060:   //PAABB', '^TAABB // will not work
14061:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14062:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14063:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14064:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14065:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14066:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14067:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14068:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14069:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14070:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14071:  Function BBMinX( const c : THmgBoundingBox) : Single
14072:  Function BBMaxX( const c : THmgBoundingBox) : Single
14073:  Function BBMinY( const c : THmgBoundingBox) : Single
14074:  Function BBMaxY( const c : THmgBoundingBox) : Single
14075:  Function BBMinZ( const c : THmgBoundingBox) : Single
14076:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14077:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14078:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14079:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14080:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14081:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14082:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14083:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14084:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14085:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14086:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14087:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14088:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14089:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14090:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14091:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14092:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14093:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14094:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14095:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14096:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14097:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14098:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14099:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14100:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14101:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14102:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14103:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
14104:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14105:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14106:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14107:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14108:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14109:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
       viewportSizeY:Int):TClipRect
14110: end;
14111:
14112: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14113: begin
14114:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14115:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14116:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14117:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14118:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14119:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14120:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14121:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14122:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14123:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14124:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14125:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14126:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14127:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14128:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14129:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14130:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14131:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14132:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
```

```
14133:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14134:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14135:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14136:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14137:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14138:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14139: end;
14140:
14141: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14142: begin
14143:  'EPSILON','Single').setExtended( 1e-40);
14144:  'EPSILON2','Single').setExtended( 1e-30);  }
14145:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14146:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14147:  THmgPlane', 'TVector
14148:   TDoubleHmgPlane', 'THomogeneousDblVector
14149:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14150:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14151:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14152:   TSingleArray', 'array of Single
14153:   TTransformations','array [0..15] of Single)
14154:   TPackedRotationMatrix','array [0..2] of Smallint)
14155:   TVertex', 'TAffineVector
14156:   //TVectorGL', 'THomogeneousFltVector
14157:   //TMatrixGL', 'THomogeneousFltMatrix
14158:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14159:  Function glTexPointMake( const s, t : Single) : TTexPoint
14160:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14161:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14162:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14163:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14164:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14165:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14166:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14167:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14168:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14169:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14170:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14171:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14172:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14173:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14174:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14175:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14176:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14177:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14178:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14179:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14180:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14181:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14182:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14183:  Procedure glRstVector( var v : TAffineVector);
14184:  Procedure glRstVector1( var v : TVectorGL);
14185:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14186:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14187:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14188:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14189:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14190:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14191:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14192:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14193:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14194:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14195:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14196:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14197:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Int;dest:PTexPointArray);
14198:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14199:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
      PAffineVectorArray);
14200:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14201:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14202:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14203:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14204:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14205:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14206:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14207:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14208:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14209:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14210:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14211:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14212:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14213:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14214:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14215:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14216:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14217:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14218:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
```

```
14219:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14220:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14221:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14222:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14223:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14224:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14225:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14226:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14227:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14228:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14229:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14230:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14231:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14232:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14233:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14234:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14235:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14236:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14237:  Function glLerp( const start, stop, t : Single) : Single
14238:  Function glAngleLerp( start, stop, t : Single) : Single
14239:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14240:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14241:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14242:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14243:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14244:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14245:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14246:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14247: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14248: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
      PAffineVectorArray);
14249:  Function glVectorLength( const x, y : Single) : Single;
14250:  Function glVectorLength1( const x, y, z : Single) : Single;
14251:  Function glVectorLength2( const v : TAffineVector) : Single;
14252:  Function glVectorLength3( const v : TVectorGL) : Single;
14253:  Function glVectorLength4( const v : array of Single) : Single;
14254:  Function glVectorNorm( const x, y : Single) : Single;
14255:  Function glVectorNorm1( const v : TAffineVector) : Single;
14256:  Function glVectorNorm2( const v : TVectorGL) : Single;
14257:  Function glVectorNorm3( var V : array of Single) : Single;
14258:  Procedure glNormalizeVector( var v : TAffineVector);
14259:  Procedure glNormalizeVector1( var v : TVectorGL);
14260:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14261:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14262: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14263:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14264:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14265:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14266:  Procedure glNegateVector( var V : TAffineVector);
14267:  Procedure glNegateVector2( var V : TVectorGL);
14268:  Procedure glNegateVector3( var V : array of Single);
14269:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14270:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14271:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14272:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14273:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14274:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14275:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14276:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14277:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14278:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14279:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14280:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14281:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14282:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14283:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14284:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14285:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14286:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14287:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14288:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14289:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14290:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14291:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14292:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14293:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14294:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14295:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14296:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14297:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14298:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14299:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14300:  Procedure glAbsVector( var v : TVectorGL);
14301:  Procedure glAbsVector1( var v : TAffineVector);
14302:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14303:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14304:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14305:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14306:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
```

```
14307:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14308:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14309:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14310:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14311:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14312:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14313:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14314:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14315:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14316:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14317:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14318:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14319:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14320:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14321:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14322:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14323:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14324:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14325:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14326:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14327:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14328:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14329:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14330:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14331:  Procedure glAdjointMatrix( var M : TMatrixGL);
14332:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14333:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14334:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14335:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14336:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14337:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14338:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14339:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14340:  Procedure glInvertMatrix( var M : TMatrixGL);
14341:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14342:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14343:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14344:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14345:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14346:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14347:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14348:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14349:  Procedure glNormalizePlane( var plane : THmgPlane)
14350:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14351:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14352:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14353:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14354:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14355:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14356:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14357:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14358:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14359:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14360:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14361:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14362:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14363:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14364:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14365:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14366:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14367:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14368:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14369:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14370:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14371:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14372:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14373:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14374:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14375:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14376:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14377:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14378:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14379:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14380:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14381:  Function glLnXP1( X : Extended) : Extended
14382:  Function glLog10( X : Extended) : Extended
14383:  Function glLog2( X : Extended) : Extended;
14384:  Function glLog21( X : Single) : Single;
14385:  Function glLogN( Base, X : Extended) : Extended
14386:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14387:  Function glPower( const Base, Exponent : Single) : Single;
14388:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14389:  Function glDegToRad( const Degrees : Extended) : Extended;
14390:  Function glDegToRad1( const Degrees : Single) : Single;
14391:  Function glRadToDeg( const Radians : Extended) : Extended;
14392:  Function glRadToDeg1( const Radians : Single) : Single;
14393:  Function glNormalizeAngle( angle : Single) : Single
14394:  Function glNormalizeDegAngle( angle : Single) : Single
```

```
14395:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14396:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14397:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14398:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14399:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14400:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14401:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14402:  Function glArcCos( const X : Extended) : Extended;
14403:  Function glArcCos1( const x : Single) : Single;
14404:  Function glArcSin( const X : Extended) : Extended;
14405:  Function glArcSin1( const X : Single) : Single;
14406:  Function glArcTan21( const Y, X : Extended) : Extended;
14407:  Function glArcTan21( const Y, X : Single) : Single;
14408:  Function glFastArcTan2( y, x : Single) : Single
14409:  Function glTan( const X : Extended) : Extended;
14410:  Function glTan1( const X : Single) : Single;
14411:  Function glCoTan( const X : Extended) : Extended;
14412:  Function glCoTan1( const X : Single) : Single;
14413:  Function glSinh( const x : Single) : Single;
14414:  Function glSinh1( const x : Double) : Double;
14415:  Function glCosh( const x : Single) : Single;
14416:  Function glCosh1( const x : Double) : Double;
14417:  Function glRSqrt( v : Single) : Single
14418:  Function glRLength( x, y : Single) : Single
14419:  Function glISqrt( i : Integer) : Integer
14420:  Function glILength( x, y : Integer) : Integer;
14421:  Function glILength1( x, y, z : Integer) : Integer;
14422:  Procedure glRegisterBasedExp
14423:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14424:  Function glRoundInt( v : Single) : Single;
14425:  Function glRoundInt1( v : Extended) : Extended;
14426:  Function glTrunc( v : Single) : Integer;
14427:  Function glTrunc64( v : Extended) : Int64;
14428:  Function glInt( v : Single) : Single;
14429:  Function glInt1( v : Extended) : Extended;
14430:  Function glFrac( v : Single) : Single;
14431:  Function glFrac1( v : Extended) : Extended;
14432:  Function glRound( v : Single) : Integer;
14433:  Function glRound64( v : Single) : Int64;
14434:  Function glRound641( v : Extended) : Int64;
14435:  Function glTrunc( X : Extended) : Int64
14436:  Function glRound( X : Extended) : Int64
14437:  Function glFrac( X : Extended) : Extended
14438:  Function glCeil( v : Single) : Integer;
14439:  Function glCeil64( v : Extended) : Int64;
14440:  Function glFloor( v : Single) : Integer;
14441:  Function glFloor64( v : Extended) : Int64;
14442:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14443:  Function glSign( x : Single) : Integer
14444:  Function glIsInRange( const x, a, b : Single) : Boolean;
14445:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14446:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14447:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14448:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14449:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14450:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14451:  Function glMinFloat3( const v1, v2 : Single) : Single;
14452:  Function glMinFloat4( const v : array of Single) : Single;
14453:  Function glMinFloat5( const v1, v2 : Double) : Double;
14454:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14455:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14456:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14457:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14458:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14459:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14460:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14461:  Function glMaxFloat2( const v : array of Single) : Single;
14462:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14463:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14464:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14465:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14466:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14467:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14468:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14469:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14470:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14471:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14472:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14473:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14474:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14475:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14476:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14477:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14478:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14479:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14480:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14481:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14482:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14483:  Function glMinXYZComponent( const v : TVectorGL) : Single;
```

```
14484:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14485:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14486:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14487:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14488:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14489:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14490:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14491:  Procedure glSortArrayAscending( var a : array of Extended)
14492:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14493:  Function glClampValue1( const aValue, aMin : Single) : Single;
14494:  Function glGeometryOptimizationMode : String
14495:  Procedure glBeginFPUOnlySection
14496:  Procedure glEndFPUOnlySection
14497:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14498:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14499:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14500:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14501:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14502:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14503:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14504:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14505:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14506:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14507:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14508:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14509:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14510:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14511:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14512:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14513:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14514:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14515:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14516:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14517:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14518:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14519:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14520:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14521:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14522:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14523:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14524:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14525:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14526:  'cPI','Single').setExtended( 3.141592654);
14527:  'cPIdiv180','Single').setExtended( 0.017453292);
14528:  'c180divPI','Single').setExtended( 57.29577951);
14529:  'c2PI','Single').setExtended( 6.283185307);
14530:  'cPIdiv2','Single').setExtended( 1.570796326);
14531:  'cPIdiv4','Single').setExtended( 0.785398163);
14532:  'c3PIdiv4','Single').setExtended( 2.35619449);
14533:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14534:  'cInv360','Single').setExtended( 1 / 360);
14535:  'c180','Single').setExtended( 180);
14536:  'c360','Single').setExtended( 360);
14537:  'cOneHalf','Single').setExtended( 0.5);
14538:  'cLn10','Single').setExtended( 2.302585093);
14539:  {'MinSingle','Extended').setExtended( 1.5e-45);
14540:  'MaxSingle','Extended').setExtended( 3.4e+38);
14541:  'MinDouble','Extended').setExtended( 5.0e-324);
14542:  'MaxDouble','Extended').setExtended( 1.7e+308);
14543:  'MinExtended','Extended').setExtended( 3.4e-4932);
14544:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14545:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14546:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14547: end;
14548:
14549: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14550: begin
14551:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14552:   (FindClass('TOBJECT'),'TFaceGroups
14553:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14554:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14555:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14556:   SIRegister_TBaseMeshObject(CL);
14557:   (FindClass('TOBJECT'),'TSkeletonFrameList
14558:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14559:   SIRegister_TSkeletonFrame(CL);
14560:   SIRegister_TSkeletonFrameList(CL);
14561:   (FindClass('TOBJECT'),'TSkeleton
14562:   (FindClass('TOBJECT'),'TSkeletonBone
14563:   SIRegister_TSkeletonBoneList(CL);
14564:   SIRegister_TSkeletonRootBoneList(CL);
14565:   SIRegister_TSkeletonBone(CL);
14566:   (FindClass('TOBJECT'),'TSkeletonColliderList
14567:   SIRegister_TSkeletonCollider(CL);
```

```
14568:   SIRegister_TSkeletonColliderList(CL);
14569:   (FindClass('TOBJECT'),'TGLBaseMesh
14570:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14571:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14572:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14573:    +'QuaternionList; end
14574:   SIRegister_TSkeleton(CL);
14575:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14576:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14577:   SIRegister_TMeshObject(CL);
14578:   SIRegister_TMeshObjectList(CL);
14579:   //TMeshObjectListClass', 'class of TMeshObjectList
14580:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14581:   SIRegister_TMeshMorphTarget(CL);
14582:   SIRegister_TMeshMorphTargetList(CL);
14583:   SIRegister_TMorphableMeshObject(CL);
14584:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14585:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14586:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14587:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14588:   SIRegister_TSkeletonMeshObject(CL);
14589:   SIRegister_TFaceGroup(CL);
14590:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14591:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14592:   SIRegister_TFGVertexIndexList(CL);
14593:   SIRegister_TFGVertexNormalTexIndexList(CL);
14594:   SIRegister_TFGIndexTexCoordList(CL);
14595:   SIRegister_TFaceGroups(CL);
14596:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14597:   SIRegister_TVectorFile(CL);
14598:   //TVectorFileClass', 'class of TVectorFile
14599:   SIRegister_TGLGLSMVectorFile(CL);
14600:   SIRegister_TGLBaseMesh(CL);
14601:   SIRegister_TGLFreeForm(CL);
14602:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14603:   TGLActorOptions', 'set of TGLActorOption
14604:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14605:   (FindClass('TOBJECT'),'TGLActor
14606:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14607:   SIRegister_TActorAnimation(CL);
14608:   TActorAnimationName', 'String
14609:   SIRegister_TActorAnimations(CL);
14610:   SIRegister_TGLBaseAnimationControler(CL);
14611:   SIRegister_TGLAnimationControler(CL);
14612:   TActorFrameInterpolation', '( afpNone, afpLinear )
14613:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14614:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14615:   SIRegister_TGLActor(CL);
14616:   SIRegister_TVectorFileFormat(CL);
14617:   SIRegister_TVectorFileFormatsList(CL);
14618:   (FindClass('TOBJECT'),'EInvalidVectorFile
14619:  Function GetVectorFileFormats : TVectorFileFormatsList
14620:  Function VectorFileFormatsFilter : String
14621:  Function VectorFileFormatsSaveFilter : String
14622:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14623:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14624:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14625: end;
14626:
14627: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14628: begin
14629:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14630:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14631:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14632:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14633:   SIRegister_TOleStream(CL);
14634:   (FindClass('TOBJECT'),'TConnectionPoints
14635:   TConnectionKind', '( ckSingle, ckMulti )
14636:   SIRegister_TConnectionPoint(CL);
14637:   SIRegister_TConnectionPoints(CL);
14638:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14639:   (FindClass('TOBJECT'),'TActiveXControlFactory
14640:   SIRegister_TActiveXControl(CL);
14641:   //TActiveXControlClass', 'class of TActiveXControl
14642:   SIRegister_TActiveXControlFactory(CL);
14643:   SIRegister_TActiveFormControl(CL);
14644:   SIRegister_TActiveForm(CL);
14645:   //TActiveFormClass', 'class of TActiveForm
14646:   SIRegister_TActiveFormFactory(CL);
14647:   (FindClass('TOBJECT'),'TPropertyPageImpl
14648:   SIRegister_TPropertyPage(CL);
14649:   //TPropertyPageClass', 'class of TPropertyPage
14650:   SIRegister_TPropertyPageImpl(CL);
14651:   SIRegister_TActiveXPropertyPage(CL);
14652:   SIRegister_TActiveXPropertyPageFactory(CL);
14653:   SIRegister_TCustomAdapter(CL);
14654:   SIRegister_TAdapterNotifier(CL);
14655:   SIRegister_IFontAccess(CL);
14656:   SIRegister_TFontAdapter(CL);
```

```
14657:   SIRegister_IPictureAccess(CL);
14658:   SIRegister_TPictureAdapter(CL);
14659:   SIRegister_TOleGraphic(CL);
14660:   SIRegister_TStringsAdapter(CL);
14661:   SIRegister_TReflectorWindow(CL);
14662:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14663:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14664:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14665:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14666:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14667:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14668:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14669:  Function ParkingWindow : HWND
14670: end;
14671:
14672: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14673: begin
14674:  // TIp6Bytes = array [0..15] of Byte;
14675: {:binary form of IPv6 adress (for string conversion routines)}
14676:  // TIp6Words = array [0..7] of Word;
14677:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14678:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14679:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14680:  Function synaIsIP( const Value : string) : Boolean');
14681:  Function synaIsIP6( const Value : string) : Boolean');
14682:  Function synaIPToID( Host : string) : Ansistring');
14683:  Function synaStrToIp6( value : string) : TIp6Bytes');
14684:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14685:  Function synaStrToIp( value : string) : integer');
14686:  Function synaIpToStr( value : integer) : string');
14687:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14688:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14689:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14690:  Function xStrToIP( const Value : String) : TIPAdr');
14691:  Function xIPToStr( const Adresse : TIPAdr) : String');
14692:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14693:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14694: end;
14695:
14696: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14697: begin
14698:  AddTypeS('TSpecials', 'set of Char');
14699:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14700:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14701:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14702:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14703:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14704:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14705:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14706:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14707:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14708:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14709:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14710:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14711:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14712:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14713:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14714:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14715:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
14716:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14717:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14718:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14719:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14720:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14721:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14722:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14723:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14724:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14725:  Function synCrc32( const Value : AnsiString) : Integer');
14726:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14727:  Function Crc16( const Value : AnsiString) : Word');
14728:  Function synMD5( const Value : AnsiString) : AnsiString');
14729:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14730:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14731:  Function synSHA1( const Value : AnsiString) : AnsiString');
14732:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14733:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14734:  Function synMD4( const Value : AnsiString) : AnsiString');
14735: end;
14736:
14737: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14738: begin
14739:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14740:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14741:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14742:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14743:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14744:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14745:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
```

```
14746:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14747:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14748:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14749:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14750:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14751:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14752:    +', CP864, CP865, CP869, CP1125 )');
14753:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14754:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14755:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
      TransformTable : array of Word) : AnsiString');
14756:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
      TransformTable : array of Word; Translit : Boolean) : AnsiString');
14757:  Function GetCurCP : TMimeChar');
14758:  Function GetCurOEMCP : TMimeChar');
14759:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14760:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14761:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14762:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14763:  Function GetBOM( Value : TMimeChar) : AnsiString');
14764:  Function StringToWide( const Value : AnsiString) : WideString');
14765:  Function WideToString( const Value : WideString) : AnsiString');
14766: end;
14767:
14768: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14769: begin
14770:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14771:  Procedure WakeOnLan( MAC, IP : string)');
14772:  Function GetDNS : string');
14773:  Function GetIEProxy( protocol : string) : TProxySetting');
14774:  Function GetLocalIPs : string');
14775: end;
14776:
14777:
14778: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14779: begin
14780:  AddConstantN('synCR','Char #$0d);
14781:  Const('synLF','Char #$0a);
14782:  Const('cSerialChunk','LongInt'( 8192);
14783:  Const('LockfileDirectory','String '/var/lock');
14784:  Const('PortIsClosed','LongInt'( - 1);
14785:  Const('ErrAlreadyOwned','LongInt'( 9991);
14786:  Const('ErrAlreadyInUse','LongInt'( 9992);
14787:  Const('ErrWrongParameter','LongInt'( 9993);
14788:  Const('ErrPortNotOpen','LongInt'( 9994);
14789:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14790:  Const('ErrMaxBuffer','LongInt'( 9996);
14791:  Const('ErrTimeout','LongInt'( 9997);
14792:  Const('ErrNotRead','LongInt'( 9998);
14793:  Const('ErrFrame','LongInt'( 9999);
14794:  Const('ErrOverrun','LongInt'( 10000);
14795:  Const('ErrRxOver','LongInt'( 10001);
14796:  Const('ErrRxParity','LongInt'( 10002);
14797:  Const('ErrTxFull','LongInt'( 10003);
14798:  Const('dcb_Binary','LongWord')( $00000001);
14799:  Const('dcb_ParityCheck','LongWord')( $00000002);
14800:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14801:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14802:  Const('dcb_DtrControlMask','LongWord')( $00000030);
14803:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14804:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14805:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14806:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14807:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14808:  Const('dcb_OutX','LongWord')( $00000100);
14809:  Const('dcb_InX','LongWord')( $00000200);
14810:  Const('dcb_ErrorChar','LongWord')( $00000400);
14811:  Const('dcb_NullStrip','LongWord')( $00000800);
14812:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14813:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14814:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14815:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14816:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14817:  Const('dcb_AbortOnError','LongWord')( $00004000);
14818:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14819:  Const('synSB1','LongInt'( 0);
14820:  Const('SB1andHalf','LongInt'( 1);
14821:  Const('synSB2','LongInt'( 2);
14822:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14823:  Const('CS7fix','LongWord')( $0000020);
14824:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14825:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14826:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14827:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14828:   //AddTypeS('PDCB', '^TDCB // will not work');
14829:  //Const('MaxRates','LongInt'( 18);
14830:  //Const('MaxRates','LongInt'( 30);
14831:  //Const('MaxRates','LongInt'( 19);
14832:  Const('O_SYNC','LongWord')( $0080);
```

```
14833:  Const('synOK','LongInt'( 0);
14834:  Const('synErr','LongInt'( integer ( - 1 ));
14835:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
       HR_WriteCount, HR_Wait )');
14836:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14837:   SIRegister_ESynaSerError(CL);
14838:   SIRegister_TBlockSerial(CL);
14839:  Function GetSerialPortNames : string');
14840: end;
14841:
14842: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14843: begin
14844:  Const('DLLIconvName','String 'libiconv.so');
14845:  Const('DLLIconvName','String 'iconv.dll');
14846:   AddTypeS('size_t', 'Cardinal');
14847:   AddTypeS('iconv_t', 'Integer');
14848:   //AddTypeS('iconv_t', 'Pointer');
14849:   AddTypeS('argptr', 'iconv_t');
14850:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14851:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14852:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14853:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14854:  Function SynaIconvClose( var cd : iconv_t) : integer');
14855:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14856:  Function IsIconvloaded : Boolean');
14857:  Function InitIconvInterface : Boolean');
14858:  Function DestroyIconvInterface : Boolean');
14859:  Const('ICONV_TRIVIALP','LongInt'( 0);
14860:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14861:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14862:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14863:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14864: end;
14865:
14866: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14867: begin
14868:  Const 'ICMP_ECHO','LongInt'( 8);
14869:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14870:  Const('ICMP_UNREACH','LongInt'( 3);
14871:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14872:  Const('ICMP6_ECHO','LongInt'( 128);
14873:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14874:  Const('ICMP6_UNREACH','LongInt'( 1);
14875:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14876:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14877:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14878:   SIRegister_TPINGSend(CL);
14879:  Function PingHost( const Host : string) : Integer');
14880:  Function TraceRouteHost( const Host : string) : string');
14881: end;
14882:
14883: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14884: begin
14885:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14886:  Const('synASN1_INT','LongWord')( $02);
14887:  Const('synASN1_OCTSTR','LongWord')( $04);
14888:  Const('synASN1_NULL','LongWord')( $05);
14889:  Const('synASN1_OBJID','LongWord')( $06);
14890:  Const('synASN1_ENUM','LongWord')( $0a);
14891:  Const('synASN1_SEQ','LongWord')( $30);
14892:  Const('synASN1_SETOF','LongWord')( $31);
14893:  Const('synASN1_IPADDR','LongWord')( $40);
14894:  Const('synASN1_COUNTER','LongWord')( $41);
14895:  Const('synASN1_GAUGE','LongWord')( $42);
14896:  Const('synASN1_TIMETICKS','LongWord')( $43);
14897:  Const('synASN1_OPAQUE','LongWord')( $44);
14898:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14899:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14900:  Function synASNEncLen( Len : Integer) : AnsiString');
14901:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14902:  Function synASNEncInt( Value : Integer) : AnsiString');
14903:  Function synASNEncUInt( Value : Integer) : AnsiString');
14904:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14905:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14906:  Function synMibToId( Mib : String) : AnsiString');
14907:  Function synIdToMib( const Id : AnsiString) : String');
14908:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14909:  Function ASNdump( const Value : AnsiString) : AnsiString');
14910:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14911:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14912: end;
14913:
14914: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14915: begin
14916:  Const('cLDAPProtocol','String '389');
14917:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14918:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14919:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14920:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
```

```
14921:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14922:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14923:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14924:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14925:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14926:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14927:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14928:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14929:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14930:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14931:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14932:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14933:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14934:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14935:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14936:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14937:   SIRegister_TLDAPAttribute(CL);
14938:   SIRegister_TLDAPAttributeList(CL);
14939:   SIRegister_TLDAPResult(CL);
14940:   SIRegister_TLDAPResultList(CL);
14941:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14942:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14943:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14944:   SIRegister_TLDAPSend(CL);
14945:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14946: end;
14947:
14948:
14949: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14950: begin
14951:  Const('cSysLogProtocol','String '514');
14952:  Const('FCL_Kernel','LongInt'( 0);
14953:  Const('FCL_UserLevel','LongInt'( 1);
14954:  Const('FCL_MailSystem','LongInt'( 2);
14955:  Const('FCL_System','LongInt'( 3);
14956:  Const('FCL_Security','LongInt'( 4);
14957:  Const('FCL_Syslogd','LongInt'( 5);
14958:  Const('FCL_Printer','LongInt'( 6);
14959:  Const('FCL_News','LongInt'( 7);
14960:  Const('FCL_UUCP','LongInt'( 8);
14961:  Const('FCL_Clock','LongInt'( 9);
14962:  Const('FCL_Authorization','LongInt'( 10);
14963:  Const('FCL_FTP','LongInt'( 11);
14964:  Const('FCL_NTP','LongInt'( 12);
14965:  Const('FCL_LogAudit','LongInt'( 13);
14966:  Const('FCL_LogAlert','LongInt'( 14);
14967:  Const('FCL_Time','LongInt'( 15);
14968:  Const('FCL_Local0','LongInt'( 16);
14969:  Const('FCL_Local1','LongInt'( 17);
14970:  Const('FCL_Local2','LongInt'( 18);
14971:  Const('FCL_Local3','LongInt'( 19);
14972:  Const('FCL_Local4','LongInt'( 20);
14973:  Const('FCL_Local5','LongInt'( 21);
14974:  Const('FCL_Local6','LongInt'( 22);
14975:  Const('FCL_Local7','LongInt'( 23);
14976:   Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
14977:   SIRegister_TSyslogMessage(CL);
14978:   SIRegister_TSyslogSend(CL);
14979:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
       Content:string):Boolean;
14980: end;
14981:
14982:
14983: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14984: begin
14985:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14986:   SIRegister_TMessHeader(CL);
14987:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14988:   SIRegister_TMimeMess(CL);
14989: end;
14990:
14991: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14992: begin
14993:   (FindClass('TOBJECT'),'TMimePart');
14994:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
14995:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14996:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14997:   SIRegister_TMimePart(CL);
14998:  Const('MaxMimeType','LongInt'( 25);
14999:  Function GenerateBoundary : string');
15000: end;
15001:
15002: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
15003: begin
15004:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
15005:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
15006:  Function NeedInline( const Value : AnsiString) : boolean');
15007:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15008:  Function InlineCode( const Value : string) : string');
```

```
15009:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15010:  Function InlineEmail( const Value : string) : string');
15011: end;
15012:
15013: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15014: begin
15015:  Const('cFtpProtocol','String '21');
15016:  Const('cFtpDataProtocol','String '20');
15017:  Const('FTP_OK','LongInt'( 255);
15018:  Const('FTP_ERR','LongInt'( 254);
15019:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15020:   SIRegister_TFTPListRec(CL);
15021:   SIRegister_TFTPList(CL);
15022:   SIRegister_TFTPSend(CL);
15023:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15024:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15025:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
       ToPort, ToFile, ToUser, ToPass : string) : Boolean');
15026: end;
15027:
15028: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15029: begin
15030:  Const('cHttpProtocol','String '80');
15031:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15032:   SIRegister_THTTPSend(CL);
15033:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
15034:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
15035:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
15036:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
15037:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
       ResultData:TStrings):Bool;
15038: end;
15039:
15040: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15041: begin
15042:  Const('cSmtpProtocol','String '25');
15043:   SIRegister_TSMTPSend(CL);
15044:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
       Passw:string):Bool;
15045:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15046:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
       Username, Password : string):Boolean');
15047: end;
15048:
15049: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15050: begin
15051:  Const('cSnmpProtocol','String '161');
15052:  Const('cSnmpTrapProtocol','String '162');
15053:  Const('SNMP_V1','LongInt'( 0);
15054:  Const('SNMP_V2C','LongInt'( 1);
15055:  Const('SNMP_V3','LongInt'( 3);
15056:  Const('PDUGetRequest','LongWord')( $A0);
15057:  Const('PDUGetNextRequest','LongWord')( $A1);
15058:  Const('PDUGetResponse','LongWord')( $A2);
15059:  Const('PDUSetRequest','LongWord')( $A3);
15060:  Const('PDUTrap','LongWord')( $A4);
15061:  Const('PDUGetBulkRequest','LongWord')( $A5);
15062:  Const('PDUInformRequest','LongWord')( $A6);
15063:  Const('PDUTrapV2','LongWord')( $A7);
15064:  Const('PDUReport','LongWord')( $A8);
15065:  Const('ENoError',LongInt 0;
15066:  Const('ETooBig','LongInt')( 1);
15067:  Const('ENoSuchName','LongInt'( 2);
15068:  Const('EBadValue','LongInt'( 3);
15069:  Const('EReadOnly','LongInt'( 4);
15070:  Const('EGenErr','LongInt'( 5);
15071:  Const('ENoAccess','LongInt'( 6);
15072:  Const('EWrongType','LongInt'( 7);
15073:  Const('EWrongLength','LongInt'( 8);
15074:  Const('EWrongEncoding','LongInt'( 9);
15075:  Const('EWrongValue','LongInt'( 10);
15076:  Const('ENoCreation','LongInt'( 11);
15077:  Const('EInconsistentValue','LongInt'( 12);
15078:  Const('EResourceUnavailable','LongInt'( 13);
15079:  Const('ECommitFailed','LongInt'( 14);
15080:  Const('EUndoFailed','LongInt'( 15);
15081:  Const('EAuthorizationError','LongInt'( 16);
15082:  Const('ENotWritable','LongInt'( 17);
15083:  Const('EInconsistentName','LongInt'( 18);
15084:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15085:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15086:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15087:   SIRegister_TSNMPMib(CL);
15088:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15089:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15090:   SIRegister_TSNMPRec(CL);
15091:   SIRegister_TSNMPSend(CL);
15092:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15093:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
```

```
15094:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15095:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15096:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15097:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
        Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15098:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
        Integer; const MIBName, MIBValue : TStringList) : Integer');
15099: end;
15100:
15101: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15102: begin
15103:  Function GetDomainName2: AnsiString');
15104:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15105:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15106:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15107:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15108:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15109: end;
15110:
15111: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15112: begin
15113:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15114:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15115:  Function wwStrToDate( const S : string) : boolean');
15116:  Function wwStrToTime( const S : string) : boolean');
15117:  Function wwStrToDateTime( const S : string) : boolean');
15118:  Function wwStrToTimeVal( const S : string) : TDateTime');
15119:  Function wwStrToDateVal( const S : string) : TDateTime');
15120:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15121:  Function wwStrToInt( const S : string) : boolean');
15122:  Function wwStrToFloat( const S : string) : boolean');
15123:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15124:  Function wwNextDay( Year, Month, Day : Word) : integer');
15125:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15126:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15127:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15128:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15129:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15130:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15131:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15132:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15133:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15134: end;
15135:
15136: unit uPSI_Themes;
15137: Function ThemeServices : TThemeServices');
15138: Function ThemeControl( AControl : TControl) : Boolean');
15139: Function UnthemedDesigner( AControl : TControl) : Boolean');
15140: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15141: begin
15142:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15143: end;
15144: Unit uPSC_menus;
15145:  Function StripHotkey( const Text : string) : string');
15146:  Function GetHotkey( const Text : string) : string');
15147:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15148:  Function IsAltGRPressed : boolean');
15149:
15150: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15151: begin
15152:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15153:   SIRegister_TIdIMAP4Server(CL);
15154: end;
15155:
15156: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15157: begin
15158:  'HASH_SIZE','LongInt'( 256);
15159:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15160:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15161:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15162:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15163:    +' : Integer; Value : Variant; end');
15164:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15165:   SIRegister_TVariantSymbolTable(CL);
15166: end;
15167:
15168: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15169: begin
15170:   SIRegister_TThreadLocalVariables(CL);
15171:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15172:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15173:  Function ThreadLocals : TThreadLocalVariables');
15174:  Procedure WriteDebug( sz : String)');
15175:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15176:  'UDF_FAILURE','LongInt'( 1);
15177:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15178:  CL.AddTypeS('mTByteArray', 'array of byte;');
15179:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15180:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
```

```
15181:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15182:  function IsNetworkConnected: Boolean;
15183:  function IsInternetConnected: Boolean;
15184:  function IsCOMConnected: Boolean;
15185:  function IsNetworkOn: Boolean;
15186:  function IsInternetOn: Boolean;
15187:  function IsCOMOn: Boolean;
15188:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15189:  TmrProc', 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);');
15190:  Function SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT');
15191:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15192:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15193:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15194:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15195:  Function GetMenu( hWnd : HWND) : HMENU');
15196:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15197: end;
15198:
15199: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15200: begin
15201:   SIRegister_IDataBlock(CL);
15202:   SIRegister_ISendDataBlock(CL);
15203:   SIRegister_ITransport(CL);
15204:   SIRegister_TDataBlock(CL);
15205:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15206:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15207:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15208:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15209:   SIRegister_TCustomDataBlockInterpreter(CL);
15210:   SIRegister_TSendDataBlock(CL);
15211:   'CallSig','LongWord')( $D800);
15212:   'ResultSig','LongWord')( $D400);
15213:   'asMask','LongWord')( $00FF);
15214:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15215:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15216:  Procedure CheckSignature( Sig : Integer)');
15217: end;
15218:
15219: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15220: begin
15221:   //CL.AddTypeS('HINTERNET', '___Pointer');
15222:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15223:   CL.AddTypeS('HINTERNET', 'Integer');
15224:   CL.AddTypeS('HINTERNET2', '___Pointer');
15225:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15226:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15227:   CL.AddTypeS('INTERNET_PORT', 'Word');
15228:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15229:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15230:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15231:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15232:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15233:  Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
      lpUrlComponents:TURLComponents):BOOL;
15234:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
      dwUrlLength:DWORD):BOOL;
15235:  Function InternetCloseHandle(hInet : HINTERNET) : BOOL');
15236:  Function
      InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
      lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15237:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15238:  ;dwContext:DWORD):HINTERNET;
15239:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
      lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15240:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
      dwContext:DWORD):BOOL;
15241:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15242:  Function
      InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15243:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15244:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15245:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15246:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15247:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15248:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
      lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15249:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
      lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15250:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15251:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15252:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
      dwContext:DWORD):BOOL;
15253:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
      TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15254:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
      BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15255:  Function
      WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15256:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
```

```
15257:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15258:  Function
       FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15259:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15260:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15261:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15262:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15263:  Function
       FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15264:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15265:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15266:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15267:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15268:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15269:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15270:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15271:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15272:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15273:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15274:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15275:  Function
       GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
       PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15276:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15277:  Function
       GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15278:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
       PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15279:  Function
       HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15280:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
       dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15281:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15282:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15283:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15284:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15285:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15286:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15287:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15288:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
       lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15289:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
       lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15290:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15291:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15292:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15293:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15294: end;
15295:
15296: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15297: begin
15298:   AddTypeS('strCharSet', 'set of char');
15299:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15300:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15301:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15302:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15303:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15304:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15305:  Procedure strStripWhiteSpace( var s : string)');
15306:  Function strRemoveChar( str : string; removeChar : char) : string');
15307:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15308:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15309:  Function wwEqualStr( s1, s2 : string) : boolean');
15310:  Function strCount( s : string; delimeter : char) : integer');
15311:  Function strWhiteSpace : strCharSet');
15312:  Function wwExtractFileNameOnly( const FileName : string) : string');
15313:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15314:  Function strTrailing( s : string; delimeter : char) : string');
15315:  Function strPreceding( s : string; delimeter : char) : string');
15316:  Function wwstrReplace( s, Find, Replace : string) : string');
15317: end;
15318:
15319: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15320: begin
15321:   SIRegister_TRemoteDataModule(CL);
15322:   SIRegister_TCRemoteDataModule(CL);
15323:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15324:  Procedure UnregisterPooled( const ClassID : string)');
15325:  Procedure EnableSocketTransport( const ClassID : string)');
15326:  Procedure DisableSocketTransport( const ClassID : string)');
15327:  Procedure EnableWebTransport( const ClassID : string)');
15328:  Procedure DisableWebTransport( const ClassID : string)');
15329: end;
15330:
15331: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15332: begin
15333:  Function mxArcCos( x : Real) : Real');
15334:  Function mxArcSin( x : Real) : Real');
```

```
15335:  Function Comp2Str( N : Comp) : String');
15336:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15337:  Function Int2Str( N : LongInt) : String');
15338:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15339:  Function LogXY( x, y : Real) : Real');
15340:  Function Pennies2Dollars( C : Comp) : String');
15341:  Function mxPower( X : Integer; Y : Integer) : Real');
15342:  Function Real2Str( N : Real; Width, Places : integer) : String');
15343:  Function mxStr2Comp( MyString : string) : Comp');
15344:  Function mxStr2Pennies( S : String) : Comp');
15345:  Function Str2Real( MyString : string) : Real');
15346:  Function XToTheY( x, y : Real) : Real');
15347: end;
15348:
15349: //************************Cindy Functions!*****************************
15350: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15351: begin
15352:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15353:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15354:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15355:  MessagePlainText','String 'text/plain');
15356:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15357:  MessageAlterText_Html','String 'multipart/alternative');
15358:  MessageHtml_Attach','String 'multipart/mixed');
15359:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15360:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15361:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15362:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15363:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15364:  Function ForceDecodeHeader( aHeader : String) : String');
15365:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15366:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15367:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15368:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15369:  Function Get_MD5( const aFileName : string) : string');
15370:  Function Get_MD5FromString( const aString : string) : string');
15371: end;
15372:
15373: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15374: begin
15375:  Function IsFolder( SRec : TSearchrec) : Boolean');
15376:  Function isFolderReadOnly( Directory : String) : Boolean');
15377:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15378:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15379:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15380:  Function DiskFreeBytes( Drv : Char) : Int64');
15381:  Function DiskBytes( Drv : Char) : Int64');
15382:  Function GetFileBytes( Filename : String) : Int64');
15383:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15384:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15385:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15386:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15387:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15388:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15389:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15390:  SE_TCB_NAME','String 'SeTcbPrivilege');
15391:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15392:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15393:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15394:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15395:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15396:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15397:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15398:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15399:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15400:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15401:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15402:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15403:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
15404:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15405:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15406:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15407:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15408:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15409:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15410:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15411:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15412: end;
15413:
15414:
15415: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15416: begin
15417:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15418:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15419:  Function ShellGetExtensionName( FileName : String) : String');
15420:  Function ShellGetIconIndex( FileName : String) : Integer');
15421:  Function ShellGetIconHandle( FileName : String) : HIcon');
15422:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15423:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
```

```
15424:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15425:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15426:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15427:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15428:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15429:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
       WaitCloseCompletion : boolean) : Boolean');
15430:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15431:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15432:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15433:  Function GetModificationDate( Filename : String) : TDateTime');
15434:  Function GetCreationDate( Filename : String) : TDateTime');
15435:  Function GetLastAccessDate( Filename : String) : TDateTime');
15436:  Function FileDelete( Filename : String) : Boolean');
15437:  Function FileIsOpen( Filename : string) : boolean');
15438:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15439:  Function DirectoryDelete( Directory : String) : Boolean');
15440:  Function GetPrinters( PrintersList : TStrings) : Integer');
15441:  Procedure SetDefaultPrinter( PrinterName : String)');
15442:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15443:  Function WinToDosPath( WinPathName : String) : String');
15444:  Function DosToWinPath( DosPathName : String) : String');
15445:  Function cyGetWindowsVersion : TWindowsVersion');
15446:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15447:  Procedure WindowsShutDown( Restart : boolean)');
15448:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
       FileIcon:string;NumIcone:integer);
15449:  Procedure GetWindowsFonts( FontsList : TStrings)');
15450:  Function GetAvailableFilename( DesiredFileName : String): String');
15451: end;
15452:
15453: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15454: begin
15455:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15456:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15457:   Type(TStringRead', '( srFromLeft, srFromRight )');
15458:   Type(TStringReads', 'set of TStringRead');
15459:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15460:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15461:   Type(TWordsOptions', 'set of TWordsOption');
15462:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15463:   Type(TCarTypes', 'set of TCarType');
15464:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15465:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15466:  Function Char_GetType( aChar : Char) : TCarType');
15467:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15468:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15469:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15470:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15471:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15472:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15473:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15474:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15475:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15476:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15477:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15478:  Function String_Quote( Str : String) : String');
15479:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15480:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15481:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15482:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15483:  Function String_ToInt( Str : String) : Integer');
15484:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15485:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15486:  Function String_Reverse( Str : String) : String');
15487:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15488:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15489:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15490:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15491:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
       Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15492:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15493:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15494:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15495:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15496:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15497:  Function String_End( Str : String; Cars : Word) : String');
15498:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15499:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15500:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15501:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15502:  Function String_IsNumbers( Str : String) : Boolean');
15503:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15504:  Function StringToCsvCell( aStr : String) : String');
15505: end;
15506:
15507: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
```

```
15508: begin
15509:  Function LongDayName( aDate : TDate) : String');
15510:  Function LongMonthName( aDate : TDate) : String');
15511:  Function ShortYearOf( aDate : TDate) : byte');
15512:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15513:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15514:  Function SecondsToMinutes( Seconds : Integer) : Double');
15515:  Function MinutesToSeconds( Minutes : Double) : Integer');
15516:  Function MinutesToHours( Minutes : Integer) : Double');
15517:  Function HoursToMinutes( Hours : Double) : Integer');
15518:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15519:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15520:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15521:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15522:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15523:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15524:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15525:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15526:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15527: end;
15528:
15529: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15530: begin
15531:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15532:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15533:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15534:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15535:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15536:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15537:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15538:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15539:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15540:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15541:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15542:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15543:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15544:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
15545:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15546:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15547:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15548:  Procedure cyCenterControl( aControl : TControl)');
15549:  Function GetLastParent( aControl : TControl) : TWinControl');
15550:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15551:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15552: end;
15553:
15554: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15555: begin
15556:  Function TablePackTable( Tab : TTable) : Boolean');
15557:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15558:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15559:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15560:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15561:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15562:  Function TableEmptyTable( Tab : TTable) : Boolean');
15563:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15564:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15565:  Function
       TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Boolean):TTable;
15566:  Function
       TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15567:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15568: end;
15569:
15570: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15571: begin
15572:   SIRegister_TcyRunTimeDesign(CL);
15573:   SIRegister_TcyShadowText(CL);
15574:   SIRegister_TcyBgPicture(CL);
15575:   SIRegister_TcyGradient(CL);
15576:   SIRegister_tcyBevel(CL);
15577:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15578:   SIRegister_tcyBevels(CL);
15579:   SIRegister_TcyImagelistOptions(CL);
15580:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15581: end;
15582:
15583: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15584: begin
15585:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
       adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
       Maxdegrade : Byte;'+
15586:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15587:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15588:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15589:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
```

```
15590:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
        Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15591:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15592:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15593:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
        BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15594:  Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
        BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Bool;const DrawRight:Bool;const
        DrawBottom:Bool;const RoundRect:bool;
15595:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
        aState : TButtonState; Focused, Hot : Boolean)');
15596:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
        aState : TButtonState; Focused, Hot : Boolean)');
15597:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
        TColor; aState : TButtonState; Focused, Hot : Boolean)');
15598:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
        GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15599:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
        const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15600:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
        TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15601:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
        TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15602:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
        WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15603:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15604:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15605:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15606:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
        CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15607:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15608:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15609:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15610:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15611:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15612:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15613:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15614:  Procedure DrawCanvas1(Destination:TCanvas;
        DestRect:TRect;Src:TCanvas;SrcRect:TRect;TransparentColor:TColor; const aStyle:TBgStyle; const
        aPosition:TBgPosition; const IndentX : Integer; const IndentY : Integer;const IntervalX : Integer; const
        IntervalY : Integer; const RepeatX:Integer;const RepeatY:Integer);
15615:  Procedure DrawGraphic( Destination : TCanvas; DestRect : TRect; aGraphic : TGraphic; SrcRect : TRect;
        TransparentColor:TColor; const aStyle : TBgStyle; const aPosition : TBgPosition; const IndentX : Integer;
        const IndentY:Integer;const IntervalX:Integer;const IntervalY:Integer;const RepeatX:Integer;const
        RepeatY:Integer;
15616:  Procedure DrawGraphic1(Destination:TCanvas; DestRect:TRect;aGraphic : TGraphic; Transparent : Boolean;
        const aStyle : TBgStyle; const aPosition : TBgPosition; const IndentX : Integer; const IndentY : Integer;
        const  IntervalX:Integer; const IntervalY : Integer; const RepeatX:Integer; const RepeatY:Integer);');
15617:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15618:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15619:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15620:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15621:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15622:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15623:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15624:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15625:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15626:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15627:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15628:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15629:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15630:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15631:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15632:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15633: end;
15634:
15635: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15636: begin
15637:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15638:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15639:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15640:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15641:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15642:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15643:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15644:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15645:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15646:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15647:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15648:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
        bmInvertReverseFromColor);
15649:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
        rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15650:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15651:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15652: end;
15653:
```

```
15654: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15655: begin
15656:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15657:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15658:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15659:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15660:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15661:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15662:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15663:  Const SC_GROUP_IDENTIFIERW','String') '+');
15664:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15665:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15666:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15667:  Const SERVICE_INACTIVE','LongWord $00000002);
15668:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15669:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15670:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15671:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15672:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15673:  Const SERVICE_STOPPED','LongWord $00000001);
15674:  Const SERVICE_START_PENDING','LongWord $00000002);
15675:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15676:  Const SERVICE_RUNNING','LongWord $00000004);
15677:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15678:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15679:  Const SERVICE_PAUSED','LongWord $00000007);
15680:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15681:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15682:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15683:  Const SC_MANAGER_CONNECT','LongWord $0001);
15684:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15685:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15686:  Const SC_MANAGER_LOCK','LongWord $0008);
15687:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15688:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15689:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15690:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15691:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15692:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15693:  Const SERVICE_START','LongWord $0010);
15694:  Const SERVICE_STOP','LongWord $0020);
15695:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15696:  Const SERVICE_INTERROGATE','LongWord $0080);
15697:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15698:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15699:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15700:  Const SERVICE_ADAPTER','LongWord $00000004);
15701:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15702:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15703:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15704:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15705:  Const SERVICE_BOOT_START','LongWord $00000000);
15706:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15707:  Const SERVICE_AUTO_START','LongWord $00000002);
15708:  Const SERVICE_DEMAND_START','LongWord $00000003);
15709:  Const SERVICE_DISABLED','LongWord $00000004);
15710:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15711:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15712:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15713:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15714:   CL.AddTypeS('SC_HANDLE', 'THandle');
15715:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15716:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15717:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15718:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15719:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15720:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15721:   Const TServiceStatus', '_SERVICE_STATUS');
15722:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15723:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15724:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15725:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15726:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15727:   TEnumServiceStatus', 'TEnumServiceStatusA');
15728:   SC_LOCK', '___Pointer');
15729:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15730:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15731:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15732:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15733:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15734:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15735:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15736:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15737:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15738:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15739:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15740:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15741:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15742:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
```

```
15743:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15744:   TQueryServiceConfig', 'TQueryServiceConfigA');
15745:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15746:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15747:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
       dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15748:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15749:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
       dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15750:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15751:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15752:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
       TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15753:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
       lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
       : DWORD):BOOL');
15754:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
       lpcchBuffer:DWORD):BOOL');
15755:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
       lpcchBuffer:DWORD):BOOL;
15756:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15757:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15758:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15759:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15760:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
       cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15761:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15762:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15763:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15764:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15765: end;
15766:
15767: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15768: begin
15769:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
       AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
       BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15770:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
       DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15771:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15772:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
       TWinControl;
15773:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
       AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15774:  Function CreateNotifyThread(const
       FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15775: end;
15776:
15777: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15778: begin
15779:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15780:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15781:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15782:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15783:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15784:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15785:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15786:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15787:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15788:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15789:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15790:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15791:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15792:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15793:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15794:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15795:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15796:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15797:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15798:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15799:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15800:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15801:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15802:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15803:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15804:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15805:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15806:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15807:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15808:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15809:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15810:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15811:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15812:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15813:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15814:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15815:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15816:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15817:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
```

```
15818:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15819:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15820:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15821:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15822:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15823:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15824:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
       List:TStrings):Bool
15825:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15826:   FindClass('TOBJECT'),'EJclFileSummaryError');
15827:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15828:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15829:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15830:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15831:   SIRegister_TJclFilePropertySet(CL);
15832:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15833:   SIRegister_TJclFileSummary(CL);
15834:   SIRegister_TJclFileSummaryInformation(CL);
15835:   SIRegister_TJclDocSummaryInformation(CL);
15836:   SIRegister_TJclMediaFileSummaryInformation(CL);
15837:   SIRegister_TJclMSISummaryInformation(CL);
15838:   SIRegister_TJclShellSummaryInformation(CL);
15839:   SIRegister_TJclStorageSummaryInformation(CL);
15840:   SIRegister_TJclImageSummaryInformation(CL);
15841:   SIRegister_TJclDisplacedSummaryInformation(CL);
15842:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15843:   SIRegister_TJclMiscSummaryInformation(CL);
15844:   SIRegister_TJclWebViewSummaryInformation(CL);
15845:   SIRegister_TJclMusicSummaryInformation(CL);
15846:   SIRegister_TJclDRMSummaryInformation(CL);
15847:   SIRegister_TJclVideoSummaryInformation(CL);
15848:   SIRegister_TJclAudioSummaryInformation(CL);
15849:   SIRegister_TJclControlPanelSummaryInformation(CL);
15850:   SIRegister_TJclVolumeSummaryInformation(CL);
15851:   SIRegister_TJclShareSummaryInformation(CL);
15852:   SIRegister_TJclLinkSummaryInformation(CL);
15853:   SIRegister_TJclQuerySummaryInformation(CL);
15854:   SIRegister_TJclImageInformation(CL);
15855:   SIRegister_TJclJpegSummaryInformation(CL);
15856: end;
15857:
15858: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15859: begin
15860:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15861:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15862:   T8087Infinity','( icProjective, icAffine )');
15863:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15864:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15865:  Function Get8087ControlWord : Word');
15866:  Function Get8087Infinity : T8087Infinity');
15867:  Function Get8087Precision : T8087Precision');
15868:  Function Get8087Rounding : T8087Rounding');
15869:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15870:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15871:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15872:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15873:  Function Set8087ControlWord( const Control : Word) : Word');
15874:  Function ClearPending8087Exceptions : T8087Exceptions');
15875:  Function GetPending8087Exceptions : T8087Exceptions');
15876:  Function GetMasked8087Exceptions : T8087Exceptions');
15877:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15878:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15879:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15880: end;
15881:
15882: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15883: begin
15884:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15885:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15886:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
       Accept:Bool;Sorted:Bool;
15887:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15888:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15889:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15890:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15891:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15892: end;
15893:
15894: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15895: begin
15896:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15897:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15898:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15899:   type ULONG', 'Cardinal');
15900:      LPCWSTR', 'PChar');
15901:   CL.AddTypeS('LPWSTR', 'PChar');
15902:   LPSTR', 'PChar');
15903:   TBindVerb', 'ULONG');
15904:   TBindInfoF', 'ULONG');
```

```
15905:   TBindF', 'ULONG');
15906:   TBSCF', 'ULONG');
15907:   TBindStatus', 'ULONG');
15908:   TCIPStatus', 'ULONG');
15909:   TBindString', 'ULONG');
15910:   TPiFlags', 'ULONG');
15911:   TOIBdgFlags', 'ULONG');
15912:   TParseAction', 'ULONG');
15913:   TPSUAction', 'ULONG');
15914:   TQueryOption', 'ULONG');
15915:   TPUAF', 'ULONG');
15916:   TSZMFlags', 'ULONG');
15917:   TUrlZone', 'ULONG');
15918:   TUrlTemplate', 'ULONG');
15919:   TZAFlags', 'ULONG');
15920:   TUrlZoneReg', 'ULONG');
15921:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15922:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15923:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15924:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15925:  const 'CF_NULL','LongInt').SetInt( 0);
15926:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15927:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15928:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15929:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15930:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15931:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15932:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15933:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15934:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15935:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15936:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15937:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15938:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15939:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15940:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15941:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15942:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15943:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15944:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15945:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15946:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15947:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15948:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15949:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15950:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15951:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15952:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15953:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15954:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15955:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15956:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15957:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15958:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15959:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15960:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15961:   SIRegister_IPersistMoniker(CL);
15962:   SIRegister_IBindProtocol(CL);
15963:   SIRegister_IBinding(CL);
15964:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15965:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15966:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15967:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15968:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15969:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15970:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15971:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15972:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
15973:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15974:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15975:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15976:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
15977:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
15978:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
15979:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
15980:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
15981:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
15982:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
15983:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
15984:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
15985:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
15986:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
15987:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
15988:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
15989:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
15990:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
15991:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
15992:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
15993:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
```

```
15994:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
15995:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
15996:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
15997:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
15998:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
15999:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16000:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16001:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16002:  const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16003:  const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
16004:  const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
16005:  const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16006:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16007:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16008:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16009:  const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
16010:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
16011:  const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16012:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16013:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16014:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16015:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16016:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16017:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16018:  const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16019:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16020:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16021:  const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
16022:  const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
16023:  const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
16024:  const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16025:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16026:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16027:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16028:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16029:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16030:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16031:  const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
16032:  const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16033:  const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
16034:  const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16035:  const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16036:  const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16037:  const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
16038:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16039:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16040:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16041:  const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
16042:  const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
16043:  const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16044:  const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16045:  const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16046:  const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16047:  const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16048:  // PBindInfo', '^TBindInfo // will not work');
16049:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16050:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16051:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16052:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16053:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16054:   TBindInfo', '_tagBINDINFO');
16055:   BINDINFO', '_tagBINDINFO');}
16056:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16057:    +'criptor : DWORD; bInheritHandle : BOOL; end');
16058:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16059:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16060:   //PRemBindInfo', '^TRemBindInfo // will not work');
16061:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16062:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16063:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16064:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16065:    +'n; dwReserved : DWORD; end');
16066:   TRemBindInfo', '_tagRemBINDINFO');
16067:   RemBINDINFO', '_tagRemBINDINFO');}
16068:   //PRemFormatEtc', '^TRemFormatEtc // will not work');
16069:   tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16070:   TRemFormatEtc', 'tagRemFORMATETC');
16071:   RemFORMATETC', 'tagRemFORMATETC');
16072:   SIRegister_IBindStatusCallback(CL);
16073:   SIRegister_IAuthenticate(CL);
16074:   SIRegister_IHttpNegotiate(CL);
16075:   SIRegister_IWindowForBindingUI(CL);
16076:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16077:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16078:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16079:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16080:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16081:  const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16082:  const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT',LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
```

```
16083:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16084:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16085:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16086:   SIRegister_ICodeInstall(CL);
16087:   SIRegister_IWinInetInfo(CL);
16088:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16089:   SIRegister_IHttpSecurity(CL);
16090:   SIRegister_IWinInetHttpInfo(CL);
16091:   SIRegister_IBindHost(CL);
16092:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16093:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16094:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16095:  Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback) : HResult');
16096:  Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback) : HResult');
16097:  Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
       : IBindStatusCallback) : HResult');
16098:  Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
      IBindStatusCallback) : HResult');
16099:  Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
      p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16100:  Function HlinkGoBack( unk : IUnknown) : HResult');
16101:  Function HlinkGoForward( unk : IUnknown) : HResult');
16102:  Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult');
16103: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult');
16104:   SIRegister_IInternet(CL);
16105:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16106:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16107:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16108:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16109:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16110:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16111:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16112:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16113:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16114:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16115:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16116:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16117:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16118:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16119:   //POLEStrArray', '^TOLESTRArray // will not work');
16120:   SIRegister_IInternetBindInfo(CL);
16121:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16122:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16123:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16124:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16125:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16126:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16127:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16128:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16129:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16130:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16131:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16132:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16133:   //PProtocolData', '^TProtocolData // will not work');
16134:   _tagPROTOCOLDATA','record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16135:   TProtocolData', '_tagPROTOCOLDATA');
16136:   PROTOCOLDATA', '_tagPROTOCOLDATA');
16137:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink, 'IInternetProtocolSink');
16138:   SIRegister_IInternetProtocolRoot(CL);
16139:   SIRegister_IInternetProtocol(CL);
16140:   SIRegister_IInternetProtocolSink(CL);
16141:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16142:   SIRegister_IInternetSession(CL);
16143:   SIRegister_IInternetThreadSwitch(CL);
16144:   SIRegister_IInternetPriority(CL);
16145:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16146:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16147:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16148:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16149:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16150:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16151:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16152:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16153:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16154:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16155:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16156:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16157:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16158:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16159:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16160:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16161:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16162:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16163:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16164:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16165:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16166:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16167:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16168:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
```

```
16169:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16170:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16171:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16172:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16173:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16174:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16175:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16176:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16177:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16178:   SIRegister_IInternetProtocolInfo(CL);
16179:   IOInet', 'IInternet');
16180:   IOInetBindInfo', 'IInternetBindInfo');
16181:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16182:   IOInetProtocol', 'IInternetProtocol');
16183:   IOInetProtocolSink', 'IInternetProtocolSink');
16184:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16185:   IOInetSession', 'IInternetSession');
16186:   IOInetPriority', 'IInternetPriority');
16187:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16188:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16189:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16190:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16191:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16192:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16193:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16194:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16195:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16196:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16197:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16198:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16199:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16200:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16201:   //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16202:  //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16203: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16204: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16205:  //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16206:  //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16207:  //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16208:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16209:   SIRegister_IInternetSecurityMgrSite(CL);
16210:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16211:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16212:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16213:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16214:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16215:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16216:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16217:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16218:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16219:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16220:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16221:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16222:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16223:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16224:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16225:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16226:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16227:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16228:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16229:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16230:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16231:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16232:  const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16233:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16234:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16235:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16236:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16237:   SIRegister_IInternetSecurityManager(CL);
16238:   SIRegister_IInternetHostSecurityManager(CL);
16239:   SIRegister_IInternetSecurityManagerEx(CL);
16240:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16241:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16242:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16243:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16244:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16245:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16246:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16247:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16248:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
```

```
16249:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16250:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16251:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16252:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16253:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16254:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16255:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16256:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16257:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16258:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16259:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16260:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16261:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16262:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16263:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16264:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16265:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16266:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16267:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16268:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16269:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16270:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16271:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16272:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16273:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16274:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16275:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16276:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16277:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16278:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16279:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16280:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16281:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16282:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16283:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16284:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16285:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16286:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16287:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16288:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16289:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16290:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16291:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16292:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16293:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16294:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16295:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16296:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16297:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16298:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16299:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16300:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16301:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16302:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16303:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16304:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16305:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16306:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16307:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16308:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16309:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16310:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16311:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16312:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16313:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16314:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16315:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16316:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16317:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16318:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16319:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16320:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16321:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16322:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16323:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16324:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16325:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16326:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16327:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16328:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16329:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16330:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16331:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16332:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16333:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16334:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16335:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16336:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16337:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
```

```
16338:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16339:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16340:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16341:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16342:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16343:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16344:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16345:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16346:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16347:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16348:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16349:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16350:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16351:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16352:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16353:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16354:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16355:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16356:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16357:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16358:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16359:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16360:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16361:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16362:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16363:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16364:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16365:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16366:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16367:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16368:   //PZoneAttributes', '^TZoneAttributes // will not work');
16369:  _ZONEATTRIBUTES','record cbSize:ULONG;szDisplayName:array [0..260-1] of Char;szDescription:array [0..200 -
        1] of Char;szIconPath: array [0..260 - 1] of char; dwTemplateMinLevel: DWORD;  dwTemplateRecommended:
       DWORD;  dwTemplateCurrentLevel: DWORD;  dwFlags: DWORD; end');
16370:   { _ZONEATTRIBUTES = packed record
16371:     cbSize: ULONG;
16372:     szDisplayName: array [0..260 - 1] of WideChar;
16373:     szDescription: array [0..200 - 1] of WideChar;
16374:     szIconPath: array [0..260 - 1] of WideChar;
16375:     dwTemplateMinLevel: DWORD;
16376:     dwTemplateRecommended: DWORD;
16377:     dwTemplateCurrentLevel: DWORD;
16378:     dwFlags: DWORD;
16379:   end; }
16380:   TZoneAttributes', '_ZONEATTRIBUTES');
16381:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16382:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16383:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16384:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16385:   SIRegister_IInternetZoneManager(CL);
16386:   SIRegister_IInternetZoneManagerEx(CL);
16387:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16388:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16389:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16390:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16391:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16392:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16393:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16394:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16395:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16396:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16397:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16398:   TCodeBaseHold', '_tagCODEBASEHOLD');
16399:   CODEBASEHOLD', '_tagCODEBASEHOLD');
16400:   //PSoftDistInfo', '^TSoftDistInfo // will not work');
16401:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16402:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16403:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
16404:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16405:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16406:   TSoftDistInfo', '_tagSOFTDISTINFO');
16407:   SOFTDISTINFO', '_tagSOFTDISTINFO');
16408:   SIRegister_ISoftDistExt(CL);
16409:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16410:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
       dwAdvertisedVersionLS : DWORD) : HResult');
16411:   SIRegister_IDataFilter(CL);
16412:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16413:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
16414:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16415:    +'terFlags : DWORD; end');
16416:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16417:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16418:   //PDataInfo', '^TDataInfo // will not work');
16419:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16420:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16421:   TDataInfo', '_tagDATAINFO');
16422:   DATAINFO', '_tagDATAINFO');
16423:   SIRegister_IEncodingFilterFactory(CL);
```

```
16424:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16425:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16426:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16427:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16428:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16429:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16430:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16431:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16432:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16433: end;
16434:
16435: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16436: begin
16437:  Procedure reformatMemo( const m : TCustomMemo)');
16438:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16439:  Procedure MoveToTop( memo : TMemo)');
16440:  Procedure ScrollToTop( memo : TMemo)');
16441:  Function LineNumberClicked( memo : TMemo) : integer');
16442:  Function MemoClickedLine( memo : TMemo) : integer');
16443:  Function ClickedMemoLine( memo : TMemo) : integer');
16444:  Function MemoLineClicked( memo : TMemo) : integer');
16445:  Function LinePositionClicked( Memo : TMemo) : integer');
16446:  Function ClickedMemoPosition( memo : TMemo) : integer');
16447:  Function MemoPositionClicked( memo : TMemo) : integer');
16448:  Procedure AdjustGridSize( grid : TDrawGrid)');
16449:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16450:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16451:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16452:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16453:  Procedure sortstrDown( var s : string)');
16454:  Procedure sortstrUp( var s : string)');
16455:  Procedure rotatestrleft( var s : string)');
16456:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16457:  Function deblank( s : string) : string');
16458:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16459:  Procedure FreeAndClearListBox( C : TListBox);');
16460:  Procedure FreeAndClearMemo( C : TMemo);');
16461:  Procedure FreeAndClearStringList( C : TStringList);');
16462:  Function dffgetfilesize( f : TSearchrec) : int64');
16463: end;
16464:
16465: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16466: begin
16467:   CL.AddTypeS('intset', 'set of byte');
16468:   TPoint64', 'record x : int64; y : int64; end');
16469:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16470:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16471:  Function GeneratePentagon( n : integer) : integer');
16472:  Function IsPentagon( p : integer) : boolean');
16473:  Function isSquare( const N : int64) : boolean');
16474:  Function isCube( const N : int64) : boolean');
16475:  Function isPalindrome( const n : int64) : boolean;');
16476:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16477:  Function GetEulerPhi( n : int64) : int64');
16478:  Function dffIntPower( a, b : int64) : int64;');
16479:  Function IntPower1( a : extended; b : int64) : extended;');
16480:  Function gcd2( a, b : int64) : int64');
16481:  Function GCDMany( A : array of integer) : integer');
16482:  Function LCMMany( A : array of integer) : integer');
16483:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16484:  Function dffFactorial( n : int64) : int64');
16485:  Function digitcount( n : int64) : integer');
16486:  Function nextpermute( var a : array of integer) : boolean');
16487:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16488:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16489:  Function InttoBinaryStr( nn : integer) : string');
16490:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16491:  Function AngleToStr( angle : extended) : string');
16492:  Function deg2rad( deg : extended) : extended');
16493:  Function rad2deg( rad : extended) : extended');
16494:  Function GetLongToMercProjection( const long : extended) : extended');
16495:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16496:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16497:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16498:   SIRegister_TPrimes(CL);
16499:   //RIRegister_TPrimes(CL);
16500:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16501:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16502:  Function Random64( const N : Int64) : Int64;');
16503:  Procedure Randomize64');
16504:  Function Random641 : extended;');
16505:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16506: end;
16507:
16508: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16509: begin
16510:   TrealPoint', 'record x : extended; y : extended; end');
16511:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16512:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
```

```
16513:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16514:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16515:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16516:  Function realpoint( x, y : extended) : TRealPoint');
16517:  Function dist( const p1, p2 : TrealPoint) : extended');
16518:  Function intdist( const p1, p2 : TPoint) : integer');
16519:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16520:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16521:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16522:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16523:  Function GetTheta( const L : TLine) : extended;');
16524:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16525:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16526:  Procedure Extendline( var L : TLine; dist : integer);');
16527:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16528:  Function Linesintersect( line1, line2 : TLine) : boolean');
16529:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16530:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16531:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16532:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16533:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16534:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16535:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16536:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16537:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16538:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16539:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16540:  Function DegtoRad( d : extended) : extended');
16541:  Function RadtoDeg( r : extended) : extended');
16542:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16543:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16544:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16545:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16546:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16547:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16548:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16549:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16550:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16551: end;
16552:
16553:
16554: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16555: begin
16556:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16557:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16558:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16559:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16560:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16561:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16562:    +'arth : extended; Phase : extended; end');
16563:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16564:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16565:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16566:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16567:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16568:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16569:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16570:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16571:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16572:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
       ApparentRaDecl:TRPoint; end');
16573:   SIRegister_TAstronomy(CL);
16574:  Function AngleToStr( angle : extended) : string');
16575:  Function StrToAngle( s : string; var angle : extended) : boolean');
16576:  Function HoursToStr24( t : extended) : string');
16577:  Function RPoint( x, y : extended) : TRPoint');
16578:  Function getStimename( t : TDTType) : string');
16579: end;
16580:
16581: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16582: begin
16583:   TCardValue', 'Integer');
16584:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16585:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16586:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16587:   SIRegister_TCard(CL);
16588:   SIRegister_TDeck(CL);
16589: end;
16590:
16591: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16592: begin
16593:   tMethodCall', 'Procedure');
16594:   tVerboseCall', 'Procedure ( s : string)');
```

```
16595:  // PTEdge', '^TEdge // will not work');
16596:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16597:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16598:   SIRegister_TNode(CL);
16599:   SIRegister_TGraphList(CL);
16600: end;
16601:
16602: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16603: begin
16604:   ParserFloat', 'extended');
16605:   //PParserFloat', '^ParserFloat // will not work');
16606:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16607:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16608:   //POperation', '^TOperation // will not work');
16609:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16610:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16611:    +'; Token : TDFFToken; end');
16612:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16613:   (CL.FindClass('TOBJECT'),'EMathParserError');
16614:   CL.FindClass('TOBJECT'),'ESyntaxError');
16615:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16616:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16617:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16618:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16619:   (CL.FindClass('TOBJECT'),'EBadName');
16620:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16621:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16622:   SIRegister_TCustomParser(CL);
16623:   SIRegister_TExParser(CL);
16624: end;
16625:
16626:   function isService: boolean;
16627:   begin
16628:     result:= NOT(Application is TApplication);
16629:     {result:= Application is TServiceApplication;}
16630:   end;
16631:   function isApplication: boolean;
16632:   begin
16633:     result:= Application is TApplication;
16634:   end;
16635:   //SM_REMOTESESSION = $1000
16636:   function isTerminalSession: boolean;
16637:   begin
16638:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16639:   end;
16640:
16641: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16642: begin
16643:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16644:    +'String; margin_bottom : String; margin_left : String; margin_right : Strin'
16645:    +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16646:  Function cyURLEncode( const S : string) : string');
16647:  Function MakeResourceURL(const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16648:  Function MakeResourceURL1(const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;;
16649:  Function cyColorToHtml( aColor : TColor) : String');
16650:  Function HtmlToColor( aHtmlColor : String) : TColor');
16651:  //Function GetStreamEncoding( aStream : TStream) : TEncoding');
16652:  // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean');
16653:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
16654:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16655:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16656:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16657:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16658:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16659:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16660:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16661:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16662:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16663: end;
16664:
16665:
16666: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16667: begin
16668:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16669:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16670:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16671:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16672:    +'inationsRepeat, CombinationsRepeatDown )');
16673:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16674:   SIRegister_TComboSet(CL);
16675: end;
16676:
16677:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16678: begin
16679:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16680:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16681:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16682:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16683:    +'mmHandle : THandle; aString : String; userParam : Integer)');
```

```
16684:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16685:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16686:   SIRegister_TcyBaseComm(CL);
16687:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16688:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16689:  Function ValidateFileMappingName( aName : String) : String');
16690:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16691: end;
16692:
16693: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16694: begin
16695:   CL.AddTypeS('DERString', 'String');
16696:   CL.AddTypeS('DERChar', 'Char');
16697:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16698:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16699:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16700:   CL.AddTypeS('DERNString', 'String');
16701:  const DERDecimalSeparator','String').SetString( '.');
16702:  const DERDefaultChars','String')('+º@/%-
      _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16703:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16704:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16705:  Function isValidWebMailChar( aChar : Char) : Boolean');
16706:  Function isValidWebSite( aStr : String) : Boolean');
16707:  Function isValidWebMail( aStr : String) : Boolean');
16708:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16709:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16710:  Function IsDERChar( aChar : Char) : Boolean');
16711:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16712:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16713:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16714:  Function IsDERSymbols( aDERString : String) : Boolean');
16715:  Function StringToDERCharSet( aStr : String) : DERString');
16716:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16717:  Function IsDERNChar( aChar : Char) : Boolean');
16718:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16719:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
16720:  Function DERExtractWebMail( aDERStr : DERString) : String');
16721:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16722:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16723:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
      RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16724:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
      aElementsType : TElementsType) : String;');
16725:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
      Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16726: end;
16727:
16728: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16729: begin
16730:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16731:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16732:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16733:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16734:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16735:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16736:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16737:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16738:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
      RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16739:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
      PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
      PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16740:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
      SingleDestinationColor, RefreshBmp : Boolean)');
16741:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16742:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16743:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16744:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16745:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16746:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16747:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16748:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16749: end;
16750:
16751: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16752: begin
16753:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16754:   TPCMChannel', '( cMono, cStereo )');
16755:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16756:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16757:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
16758:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16759:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16760:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16761:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16762:    +'it48000Hz, Stereo16bit48000Hz )');
```

```
16763:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16764:    +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16765:    tWaveFormatEx', 'PWaveFormatEx');
16766:    HMMIO', 'Integer');
16767:   TWaveDeviceFormats', 'set of TPCMFormat');
16768:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16769:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16770:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16771:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16772:   TWaveOutOptions', 'set of TWaveOutOption');
16773:   TStreamOwnership2', '( soReference, soOwned )');
16774:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16775:  // PRawWave', '^TRawWave // will not work');
16776:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16777:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioError');
16778:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError');
16779:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation');
16780:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16781:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16782:    +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16783:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16784:    +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16785:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16786:    +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16787:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16788:    +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16789:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16790:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16791:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16792:  CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16793:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16794:  GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16795:  CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16796:  OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16797:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16798:  GetAudioFormat( FormatTag : Word) : String');
16799:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
16800:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16801:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16802:  GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16803:  InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16804:  SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16805:  ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16806:  MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
       TObject; BufferSize : DWORD) : Boolean');
16807:  ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
       dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16808:  SetPCMAudioFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
       TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16809:  Procedure SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16810:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16811:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');
16812:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) : String');
16813:  WaitForSyncObject( SyncObject : THandle; Timeout : DWORD) : DWORD');
16814:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT');
16815: end;
16816:
16817: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16818: begin
16819:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16820:  CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16821:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16822:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
16823:  'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16824:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16825:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16826:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16827:   CL.AddTypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg )');
16828:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16829:   SIRegister_TNamedPipe(CL);
16830:   SIRegister_TServerPipe(CL);
16831:   SIRegister_TClientPipe(CL);
16832:  CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD) : DWORD');
16833:  Function HasOverlappedIoCompleted( const ov : OVERLAPPED) : Boolean');
16834:  Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean):
       TOverlappedResult;
16835:  Function GetStreamAsText( stm : TStream) : string');
16836:  Procedure SetStreamAsText( const aTxt : string; stm : TStream)');
16837: end;
16838:
16839: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16840: begin
16841:  // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16842:   SIRegister_TThumbData(CL);
16843:  'PIC_BMP','LongInt').SetInt( 0);
16844:  'PIC_JPG','LongInt').SetInt( 1);
16845:  'THUMB_WIDTH','LongInt').SetInt( 60);
16846:  'THUMB_HEIGHT','LongInt').SetInt( 60);
16847:  Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean');
```

```
16848:  Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)');
16849:  Function ReadBitmap( Filehandle : Integer; Width, Height : Integer) : TBitmap');
16850:  Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap)');
16851:  Function OpenPicture( fn : string; var tp : Integer) : Integer');
16852:  Function ConvertPicture( pi : Integer; tp : Integer) : TBitmap');
16853:  Function LoadPicture( fn : string; var w, h : Integer) : TBitmap');
16854:  Function TurnBitmap( bmp : TBitmap; ang : Integer) : TBitmap');
16855:  Function RotateBitmap( Bitmap : TBitmap; Direction : Integer) : TBitmap');
16856:  Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect');
16857:  Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer) : TBitmap');
16858:  Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer)');
16859:  Procedure FindFiles( path, mask : string; items : TStringList)');
16860:  Function LetFileName( s : string) : string');
16861:  Function LetParentPath( path : string) : string');
16862:  Function AddBackSlash( path : string) : string');
16863:  Function CutBackSlash( path : string) : string');
16864: end;
16865:
16866: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16867: begin
16868:  //'BYTES','LongInt').SetInt( 1);
16869:  'KBYTES','LongInt').SetInt( 1024);
16870:  'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ));
16871:  'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ));
16872:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16873:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D}');
16874:   SIRegister_MakeComServerMethodsPublic(CL);
16875:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16876:  Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) : Boolean');
16877:  Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean)');
16878:  Function TBGetTempFolder : string');
16879:  Function TBGetTempFile : string');
16880:  Function TBGetModuleFilename : string');
16881:  Function FormatModuleVersionInfo( const aFilename : string) : string');
16882:  Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD) : string');
16883:  Function TBGetFileSize( aFile : string; aMultipleOf : Integer) : Integer');
16884:  Function FormatAttribString( aAttr : Integer) : string');
16885:  Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo) : string');
16886:  Function ShellRecycle( aWnd : HWND; aFileOrFolder : string) : Boolean');
16887:  Function IsDebuggerPresent : BOOL');
16888:  Function TBNotImplemented : HRESULT');
16889: end;
16890:
16891: procedure SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
16892: begin
16893:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16894:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16895:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16896:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16897:   //TDrivesProperty = array['A'..'Z'] of boolean;
16898:  Function TBSetSystemTime( DateTime : TDateTime; DOW : word) : boolean');
16899:  Function IsElevated : Boolean');
16900:  Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
16901:   CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16902:  Function TrimNetResource( UNC : string) : string');
16903:  Procedure GetFreeDrives( var FreeDrives : TDrivesProperty)');
16904:  Procedure GetMappedDrives( var MappedDrives : TDrivesProperty)');
16905:  Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
       UserUsage : TPasswordUsage; User : string; Comment : string) : boolean');
16906:  Function UnmapDrive( Drive : char; Force : boolean) : boolean');
16907:  Function TBIsWindowsVista : Boolean');
16908:  Procedure SetVistaFonts( const AForm : TForm)');
16909:  Procedure SetVistaContentFonts( const AFont : TFont)');
16910:  Function GetProductType( var sType : String) : Boolean');
16911:  Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16912:  Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16913:  Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar');
16914:  Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16915:  Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16916:  Function lstrlen( lpString : PChar) : Integer');
16917:  Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16918:  Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD) : BOOL');
16919: end;
16920:
16921: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16922: begin
16923:  'cLineTerminator','Char').SetString( #10);
16924:  'cLineTerminators','String').SetString( #13#10);
16925:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
16926:   SIRegister_TFixedCriticalSection(CL);
16927:   SIRegister_TMultiReadSingleWrite(CL);
16928:  CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16929:  Function GetDecimalSeparator : Char');
16930:  TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16931:  Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
       recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16932:   CL.AddTypeS('NativeInt', 'Integer');
```

```
16933:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16934:   CL.AddTypeS('NativeUInt', 'Cardinal');
16935:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16936:   //CL.AddTypeS('TBytes', 'array of Byte');
16937:   CL.AddTypeS('RawByteString', 'UnicodeString');
16938:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16939:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16940:   SIRegister_TPath(CL);
16941:   SIRegister_TFile(CL);
16942:   SIRegister_TdwsThread(CL);
16943:  Function GetSystemMilliseconds : Int64');
16944:  Function UTCDateTime : TDateTime');
16945:  Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16946:  Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16947:  Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16948:  Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
16949:  Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer;');
16950:  Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer;');
16951:  Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16952:  Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16953:  Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16954:  Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16955:  Function InterlockedIncrement( var val : Integer) : Integer');
16956:  Function InterlockedDecrement( var val : Integer) : Integer');
16957:  Procedure FastInterlockedIncrement( var val : Integer)');
16958:  Procedure FastInterlockedDecrement( var val : Integer)');
16959:  Function InterlockedExchangePointer( var target : ___Pointer; val : ___Pointer) : ___Pointer');
16960:  Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16961:  Procedure dwsOutputDebugString( const msg : UnicodeString)');
16962:  Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16963:  Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16964:  Procedure VarCopy( out dest : Variant; const src : Variant)');
16965:  Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16966:  Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16967:  Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16968:  Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16969:  Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
16970:  Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
16971:  Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
16972:  Procedure CloseFileHandle( hFile : THandle)');
16973:  Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
16974:  Function FileMove( const existing, new : UnicodeString) : Boolean');
16975:  Function dwsFileDelete( const fileName : String) : Boolean');
16976:  Function FileRename( const oldName, newName : String) : Boolean');
16977:  Function dwsFileSize( const name : String) : Int64');
16978:  Function dwsFileDateTime( const name : String) : TDateTime');
16979:  Function DirectSet8087CW( newValue : Word) : Word');
16980:  Function DirectSetMXCSR( newValue : Word) : Word');
16981:  Function TtoObject( const T: byte) : TObject');
16982:  Function TtoPointer( const T: byte) : ___Pointer');
16983:  Procedure GetMemForT(var T: byte; Size : integer)');
16984:  Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
16985: end;
16986:
16987: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
16988: begin
16989:  'IPStrSize','LongInt').SetInt( 15);
16990:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
16991:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
16992:  'ADWSBASE','LongInt').SetInt( 9000);
16993:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
16994:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
16995:   SIRegister_EApdSocketException(CL);
16996:   TWsMode', '( wsClient, wsServer )');
16997:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket');
16998:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrCode : Integer)');
16999:   SIRegister_TApdSocket(CL);
17000: end;
17001:
17002: procedure SIRegister_AdPort(CL: TPSPascalCompiler);
17003: begin
17004:   SIRegister_TApdCustomComPort(CL);
17005:   SIRegister_TApdComPort(CL);
17006:  Function ComName( const ComNumber : Word) : ShortString');
17007:  Function SearchComPort( const C : TComponent) : TApdCustomComPort');
17008: end;
17009:
17010: procedure SIRegister_PathFunc(CL: TPSPascalCompiler);
17011: begin
17012:  Function inAddBackslash( const S : String) : String');
17013:  Function PathChangeExt( const Filename, Extension : String) : String');
17014:  Function PathCharCompare( const S1, S2 : PChar) : Boolean');
17015:  Function PathCharIsSlash( const C : Char) : Boolean');
17016:  Function PathCharIsTrailByte( const S : String; const Index : Integer) : Boolean');
17017:  Function PathCharLength( const S : String; const Index : Integer) : Integer');
17018:  Function inPathCombine( const Dir, Filename : String) : String');
17019:  Function PathCompare( const S1, S2 : String) : Integer');
17020:  Function PathDrivePartLength( const Filename : String) : Integer');
17021:  Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
```

```
17022:  Function inPathExpand( const Filename : String) : String');
17023:  Function PathExtensionPos( const Filename : String) : Integer');
17024:  Function PathExtractDir( const Filename : String) : String');
17025:  Function PathExtractDrive( const Filename : String) : String');
17026:  Function PathExtractExt( const Filename : String) : String');
17027:  Function PathExtractName( const Filename : String) : String');
17028:  Function PathExtractPath( const Filename : String) : String');
17029:  Function PathIsRooted( const Filename : String) : Boolean');
17030:  Function PathLastChar( const S : String) : PChar');
17031:  Function PathLastDelimiter( const Delimiters, S : string) : Integer');
17032:  Function PathLowercase( const S : String) : String');
17033:  Function PathNormalizeSlashes( const S : String) : String');
17034:  Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17035:  Function PathPos( Ch : Char; const S : String) : Integer');
17036:  Function PathStartsWith( const S, AStartsWith : String) : Boolean');
17037:  Function PathStrNextChar( const S : PChar) : PChar');
17038:  Function PathStrPrevChar( const Start, Current : PChar) : PChar');
17039:  Function PathStrScan( const S : PChar; const C : Char) : PChar');
17040:  Function inRemoveBackslash( const S : String) : String');
17041:  Function RemoveBackslashUnlessRoot( const S : String) : String');
17042:  Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean)');
17043: end;
17044:
17045:
17046: procedure SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17047: begin
17048:  NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17049:  NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17050:  NEWREGSTR_PATH_SPECIAL_FOLDERS','String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17051:  NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17052:  NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName');
17053:  NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString');
17054:  KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17055:   //CL.AddTypeS('PLeadByteSet', '^TLeadByteSet // will not work');
17056:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17057:   SIRegister_TOneShotTimer(CL);
17058:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17059:  'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17060:  Function NewFileExists( const Name : String) : Boolean');
17061:  Function inDirExists( const Name : String) : Boolean');
17062:  Function FileOrDirExists( const Name : String) : Boolean');
17063:  Function IsDirectoryAndNotReparsePoint( const Name : String) : Boolean');
17064:  Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String');
17065:  Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17066:  Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filename:String): Boolean');
17067:  Function IniKeyExists( const Section, Key, Filename : String) : Boolean');
17068:  Function IsIniSectionEmpty( const Section, Filename : String) : Boolean');
17069:  Function SetIniString( const Section, Key, Value, Filename : String) : Boolean');
17070:  Function SetIniInt(const Section,Key:String;const Value: Longint;const Filename: String):Boolean');
17071:  Function SetIniBool(const Section,Key:String;const Value: Boolean;const Filename: String):Boolean');
17072:  Procedure DeleteIniEntry( const Section, Key, Filename : String)');
17073:  Procedure DeleteIniSection( const Section, Filename : String)');
17074:  Function GetEnv( const EnvVar : String) : String');
17075:  Function GetCmdTail : String');
17076:  Function GetCmdTailEx( StartIndex : Integer) : String');
17077:  Function NewParamCount : Integer');
17078:  Function NewParamStr( Index : Integer) : string');
17079:  Function AddQuotes( const S : String) : String');
17080:  Function RemoveQuotes( const S : String) : String');
17081:  Function inGetShortName( const LongName : String) : String');
17082:  Function inGetWinDir : String');
17083:  Function inGetSystemDir : String');
17084:  Function GetSysWow64Dir : String');
17085:  Function GetSysNativeDir( const IsWin64 : Boolean) : String');
17086:  Function inGetTempDir : String');
17087:  Function StringChange( var S : String; const FromStr, ToStr : String) : Integer');
17088:  Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17089:  Function AdjustLength( var S : String; const Res : Cardinal) : Boolean');
17090:  Function UsingWinNT : Boolean');
17091:  Function ConvertConstPercentStr( var S : String) : Boolean');
17092:  Function ConvertPercentStr( var S : String) : Boolean');
17093:  Function ConstPos( const Ch : Char; const S : String) : Integer');
17094:  Function SkipPastConst( const S : String; const Start : Integer) : Integer');
17095:  Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String) : Boolean');
17096:  Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String):Boolean;
17097:  Function RegValueExists( H : HKEY; Name : PChar) : Boolean');
17098:  Function RegCreateKeyExView(const RegView:TRegView;hKey:HKEY;lpSubKey:PChar;Reserved:DWORD;lpClass:PChar;
       dwOptions:DWORD;samDesired:REGSAM;lpSecurityAttributes:TObject;var
       phkResult:HKEY;lpdwDisposition:DWORD):Longint;
17099:  Function RegOpenKeyExView(const
       RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17100:  Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar) : Longint;
17101:  Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17102:  Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyName:PChar):Longint;
17103:  Function GetShellFolderPath( const FolderID : Integer) : String');
17104:  Function IsAdminLoggedOn : Boolean');
17105:  Function IsPowerUserLoggedOn : Boolean');
17106:  Function IsMultiByteString( const S : AnsiString) : Boolean');
17107:  Function FontExists( const FaceName : String) : Boolean');
```

```
17108:  //Procedure FreeAndNil( var Obj)');
17109:  Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT) : HMODULE');
17110:  Function GetUILanguage : LANGID');
17111:  Function RemoveAccelChar( const S : String) : String');
17112:  Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17113:  Function AddPeriod( const S : String) : String');
17114:  Function GetExceptMessage : String');
17115:  Function GetPreferredUIFont : String');
17116:  Function IsWildcard( const Pattern : String) : Boolean');
17117:  Function WildcardMatch( const Text, Pattern : PChar) : Boolean');
17118:  Function IntMax( const A, B : Integer) : Integer');
17119:  Function Win32ErrorString( ErrorCode : Integer) : String');
17120:  Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet)');
17121:  Function inCompareMem( P1, P2 : TObject; Length : Integer) : Boolean');
17122:  Function DeleteDirTree( const Dir : String) : Boolean');
17123:  Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17124:  Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT)');
17125:  // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17126:  Function inCharInSet( C : Char; const CharSet : TSysCharSet) : Boolean');
17127:  Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String) : Boolean');
17128:  Function ShutdownBlockReasonDestroy( Wnd : HWND) : Boolean');
17129:  Function TryStrToBoolean( const S : String; var BoolResult : Boolean) : Boolean');
17130:  Procedure WaitMessageWithTimeout( const Milliseconds : DWORD)');
17131:  Function MoveFileReplace(const ExistingFileName, NewFileName : String) : Boolean');
17132:  Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND)');
17133: end;
17134:
17135: procedure SIRegister_CmnFunc(CL: TPSPascalCompiler);
17136: begin
17137:   SIRegister_TWindowDisabler(CL);
17138:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17139:   TMsgBoxCallbackFunc','procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17140:  Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox)');
17141:  Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17142:  Function AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Integer');
17143:  Function MsgBoxP( const Text,Caption: PChar; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17144:  Function inMsgBox(const Text,Caption:String; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17145:  Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
       Typ:TMsgBoxType;const Buttons:Cardinal):Integer');
17146:  Procedure ReactivateTopWindow');
17147:  Procedure SetMessageBoxCaption( const Typ : TMsgBoxType; const NewCaption : PChar)');
17148:  Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean)');
17149:  Procedure SetMessageBoxCallbackFunc(const AFunc : TMsgBoxCallbackFunc; const AParam : LongInt)');
17150: end;
17151:
17152: procedure SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17153: begin
17154:   SIRegister_TImageGrabber(CL);
17155:   SIRegister_TCaptureDrivers(CL);
17156:   SIRegister_TCaptureDriver(CL);
17157: end;
17158:
17159: procedure SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17160: begin
17161:  Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
       Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean');
17162:  Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
       Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean');
17163: end;
17164:
17165: procedure SIRegister_RedirFunc(CL: TPSPascalCompiler);
17166: begin
17167:   CL.AddTypeS('TPreviousFsRedirectionState', record DidDisable : Boolean; OldValue : ___Pointer; end');
17168:  CL.AddDelphiFunction('Function AreFsRedirectionFunctionsAvailable : Boolean');
17169:  Function DisableFsRedirectionIf(const Disable:Boolean;var PreviousState:TPreviousFsRedirectionState):Boo;
17170:  Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17171:  Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17172:  Function CreateProcessRedir(const DisableFsRedir:Boolean;const lpApplicationName:PChar;const
       lpCommandLine:PChar; const lpProcessAttributes, lpThreadAttributes:PSecurityAttributes;const
       bInheritHandles:BOOL; const dwCreationFlags:DWORD;const lpEnvironment:Pointer; const
       lpCurrentDirectory:PChar; const lpStartupInfo : TStartupInfo; var
       lpProcessInformation:TProcessInformation):BOOL;
17173:  Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
       FailIfExists : BOOL) : BOOL');
17174:  Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17175:  Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17176:  Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17177:  Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
       TWin32FindData) : THandle');
17178:  Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD');
17179:  Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String) : String');
17180:  Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
       TFileVersionNumbers) : Boolean');
17181:  Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17182:  Function MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17183:  Function MoveFileExRedir(const DisableFsRedir:Boolean;const ExistingFilename,NewFilename:String;const
       Flags:DWORD):BOOL;
17184:  Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17185:  Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
```

```
17186:  Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17187:  Function SetNTFSCompressionRedir(const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17188:   SIRegister_TFileRedir(CL);
17189:   SIRegister_TTextFileReaderRedir(CL);
17190:   SIRegister_TTextFileWriterRedir(CL);
17191: end;
17192:
17193: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17194: begin
17195:   //CL.AddTypeS('LongWord', 'Cardinal');
17196:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17197:  Function Compare64( const N1, N2 : Integer64) : Integer');
17198:  Procedure Dec64( var X : Integer64; N : LongWord)');
17199:  Procedure Dec6464( var X : Integer64; const N : Integer64)');
17200:  Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord');
17201:  Function Inc64( var X : Integer64; N : LongWord) : Boolean');
17202:  Function Inc6464( var X : Integer64; const N : Integer64) : Boolean');
17203:  Function Integer64ToStr( X : Integer64) : String');
17204:  Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord');
17205:  Function Mul64( var X : Integer64; N : LongWord) : Boolean');
17206:  Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)');
17207:  Procedure Shr64( var X : Integer64; Count : LongWord)');
17208:  Function StrToInteger64( const S : String; var X : Integer64) : Boolean');
17209: end;
17210:
17211: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17212: begin
17213:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17214:   SIRegister_TSimpleStringList(CL);
17215:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle)');
17216:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17217:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17218:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17219:  // TMD5Digest = array[0..15] of Byte;
17220:   //     TSHA1Digest = array[0..19] of Byte;
17221:  Function CheckForMutexes( Mutexes : String) : Boolean');
17222:  Function CreateTempDir : String');
17223:  Function DecrementSharedCount( const RegView : TRegView; const Filename : String) : Boolean');
17224:  Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
      FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)');
17225:  //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
      DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
      TDeleteFileProc; const Param : Pointer) : Boolean');
17226:  //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
      TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
      TDetermineDefaultLanguageResult');
17227:  //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17228:  Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean');
17229:  Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17230:  Function GetComputerNameString : String');
17231:  Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17232:  Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String) : TMD5Digest');
17233:  Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest');
17234: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest');
17235:  Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest');
17236:  Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest');
17237: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest');
17238:  Function GetRegRootKeyName( const RootKey : HKEY) : String');
17239:  Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
      TotBytes:Int64):Bool;
17240:  Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
      TotalBytes: Integer64):Bool;
17241:  Function GetUserNameString : String');
17242:  Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17243:  //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
      String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
      ResultCode:Integer) : Boolean');
17244: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
      TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17245:  Procedure InternalError( const Id : String)');
17246:  Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17247:  Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
17248:  Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17249:  Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17250:  Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17251:  Procedure RaiseFunctionFailedError( const FunctionName : String)');
17252:  Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17253:  Procedure RefreshEnvironment');
17254:  Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17255:  Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17256:  Procedure UnregisterFont( const FontName, FontFilename : String)');
17257:  Function RestartComputer : Boolean');
17258:  Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17259:  Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)');
17260:  Procedure Win32ErrorMsg( const FunctionName : String)');
17261:  Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17262:  Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17263:  //from Func2
17264:  //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
      IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean; '
```

```
17265:   //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String');
17266:  Procedure RegisterTypeLibrary( const Filename : String)');
17267:  //Procedure UnregisterTypeLibrary( const Filename : String)');
17268:  //Function UnpinShellLink( const Filename : String) : Boolean');}
17269:  function getVersionInfoEx3: TOSVersionInfoEx;');
17270:  Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17271:  procedure InitOle;');
17272:  Function ExpandConst( const S : String) : String');
17273:  Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17274:  Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17275:  Function ExpandConstIfPrefixed( const S : String) : String');
17276:  Procedure LogWindowsVersion');
17277:  Function EvalCheck( const Expression : String) : Boolean');
17278: end;
17279:
17280: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17281: begin
17282:   CL.AddClassN(CL.FindClass('TOBJECT'),'TResourceDetails');
17283:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17284:   SIRegister_TResourceModule(CL);
17285:   SIRegister_TResourceDetails(CL);
17286:   SIRegister_TAnsiResourceDetails(CL);
17287:   SIRegister_TUnicodeResourceDetails(CL);
17288:  Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17289:  Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17290:  Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17291:  Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17292:  Function ResourceNameToInt( const s : string) : Integer');
17293:  Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17294: end;
17295:
17296:
17297: procedure SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17298: begin
17299:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17300:   with CL.AddClassN(CL.FindClass('TObject'),'TSimpleComPort') do begin
17301:     RegisterMethod('Constructor Create');
17302:     RegisterMethod('Procedure Free');
17303:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17304:     RegisterMethod('Procedure WriteString( const S : String)');
17305:     RegisterMethod('Procedure ReadString( var S : String)');
17306:   end;
17307:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17308:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17309:       SimpleComPort.WriteString(AsciiChar);
17310: end;
17311:
17312:
17313: procedure SIRegister_Console(CL: TPSPascalCompiler);
17314: begin
17315:  CL.AddConstantN('White','LongInt').SetInt( 15); }
17316: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17317:  CL.AddConstantN('conBW40','LongInt').SetInt( 0);
17318:  CL.AddConstantN('conCO40','LongInt').SetInt( 1);
17319:  CL.AddConstantN('conBW80','LongInt').SetInt( 2);
17320:  CL.AddConstantN('conCO80','LongInt').SetInt( 3);
17321:  CL.AddConstantN('conMono','LongInt').SetInt( 7);
17322:  CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17323:  //CL.AddConstantN('C40','').SetString( CO40);
17324:  //CL.AddConstantN('C80','').SetString( CO80);
17325:  Function conReadKey : Char');
17326:  Function conKeyPressed : Boolean');
17327:  Procedure conGotoXY( X, Y : Smallint)');
17328:  Function conWhereX : Integer');
17329:  Function conWhereY : Integer');
17330:  Procedure conTextColor( Color : Byte);');
17331:  Function conTextColor1 : Byte;');
17332:  Procedure conTextBackground( Color : Byte);');
17333:  Function conTextBackground1 : Byte;');
17334:  Procedure conTextMode( Mode : Word)');
17335:  Procedure conLowVideo');
17336:  Procedure conHighVideo');
17337:  Procedure conNormVideo');
17338:  Procedure conClrScr');
17339:  Procedure conClrEol');
17340:  Procedure conInsLine');
17341:  Procedure conDelLine');
17342:  Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17343:  Function conScreenWidth : Smallint');
17344:  Function conScreenHeight : Smallint');
17345:  Function conBufferWidth : Smallint');
17346:  Function conBufferHeight : Smallint');
17347:  procedure InitScreenMode;');
17348: end;
17349:
17350: (*----------------------------------------------------------------------------*)
17351: procedure SIRegister_testutils(CL: TPSPascalCompiler);
17352: begin
17353:   SIRegister_TNoRefCountObject(CL);
```

```
17354:  Procedure FreeObjects( List : TFPList)');
17355:  Procedure GetMethodList( AObject : TObject; AList : TStrings);');
17356:  Procedure GetMethodList1( AClass : TClass; AList : TStrings);');
17357: end;
17358:
17359: procedure SIRegister_ToolsUnit(CL: TPSPascalCompiler);
17360: begin
17361:  'MaxDataSet','LongInt').SetInt( 35);
17362:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17363:   SIRegister_TDBConnector(CL);
17364:   SIRegister_TDBBasicsTestSetup(CL);
17365:   SIRegister_TTestDataLink(CL);
17366:  'testValuesCount','LongInt').SetInt( 25);
17367:  Procedure InitialiseDBConnector');
17368:  Procedure FreeDBConnector');
17369:  Function DateTimeToTimeString( d : tdatetime) : string');
17370:  Function TimeStringToDateTime( d : String) : TDateTime');
17371: end;
17372:
17373: procedure SIRegister_fpcunit(CL: TPSPascalCompiler);
17374: begin
17375:   SIRegister_EAssertionFailedError(CL);
17376:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17377:   CL.AddTypeS('TRunMethod', 'Procedure');
17378:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTestResult');
17379:   SIRegister_TTest(CL);
17380:   SIRegister_TAssert(CL);
17381:   SIRegister_TTestFailure(CL);
17382:   SIRegister_ITestListener(CL);
17383:   SIRegister_TTestCase(CL);
17384:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17385:   SIRegister_TTestSuite(CL);
17386:   SIRegister_TTestResult(CL);
17387:  Function ComparisonMsg( const aExpected : string; const aActual : string) : string');
17388: end;
17389:
17390: procedure SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
17391: begin
17392:   TOBJECT'),'ETCPBuffer');
17393:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Intege'
17394:    +r; Head : Integer; Used : Integer; end');
17395:  'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
17396:  'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
17397:  'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
17398:  'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
17399:  Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int;
17400:  Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer)');
17401:  Procedure TCPBufferPack( var TCPBuf : TTCPBuffer)');
17402:  Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer)');
17403:  Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer)');
17404:  Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer)');
17405:  Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer) : Pointer');
17406:  Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer)');
17407:  Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Integer');
17408:  Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf: string; const Size:Integer): Integer');
17409:  Function TCPBufferRemove(var TCPBuf : TTCPBuffer;var Buf: string; const Size:Integer): Integer');
17410:  Procedure TCPBufferClear( var TCPBuf : TTCPBuffer)');
17411:  Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer) : Integer');
17412:  Function TCPBufferUsed( const TCPBuf : TTCPBuffer) : Integer');
17413:  Function TCPBufferEmpty( const TCPBuf : TTCPBuffer) : Boolean');
17414:  Function TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Integer');
17415:  Function TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer');
17416:  Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer)');
17417: end;
17418:
17419: procedure SIRegister_Glut(CL: TPSPascalCompiler);
17420: begin
17421:   //CL.AddTypeS('PInteger', '^Integer // will not work');
17422:   //CL.AddTypeS('PPChar', '^PChar // will not work');
17423:  CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3);
17424:  CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12);
17425:  CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0);
17426:  CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5);
17427:  CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6);
17428:  Procedure LoadGlut( const dll : String)');
17429:  Procedure FreeGlut');
17430: end;
17431:
17432: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17433: begin
17434:   CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17435:  Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean) : THandle');
17436: end;
17437:
17438: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17439: begin
17440:   TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )');
17441:   TTLedState', '( LedOn, LedOff, LedDisabled )');
17442:   SIRegister_TSwitchLed(CL);
```

```
17443:  //CL.AddDelphiFunction('Procedure Register');
17444: end;
17445:
17446: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17447: begin
17448:   CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17449:    +'xisting, fdOpenAlways, fdTruncateExisting )');
17450:   CL.AddTypeS('TIFileAccess', '( faRead, faWrite, faReadWrite )');
17451:   CL.AddTypeS('TIFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17452:   SIRegister_TCustomFile(CL);
17453:   SIRegister_TIFile(CL);
17454:   SIRegister_TMemoryFile(CL);
17455:   SIRegister_TTextFileReader(CL);
17456:   SIRegister_TTextFileWriter(CL);
17457:   SIRegister_TFileMapping(CL);
17458:   SIRegister_EFileError(CL);
17459: end;
17460:
17461: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17462: begin
17463:   CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17464:    +', ffaDirectory, ffaArchive, ffaAnyFile )');
17465:   CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17466:   SIRegister_TFileSearch(CL);
17467: end;
17468:
17469: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17470: begin
17471:  TRGBType', 'record RedHex : string; GreenHex : string; BlueHex :'
17472:    + string; Red : integer; Green : integer; Blue : integer; end');
17473:  Function FadeColor( aColor : Longint; aFade : integer) : Tcolor');
17474:  Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17475: end;
17476:
17477: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17478: begin
17479:  Procedure SaveOscSettings');
17480:  Procedure GetOscSettings');
17481: end;
17482:
17483: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17484: begin
17485:   TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer)');
17486:   RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17487:    FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17488:    64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17489:    Cardinal; EnterCount : Integer; ExitCount : Integer; end');
17490:   SIRegister_TcyDebug(CL);
17491: end;
17492:
17493: (*----------------------------------------------------------------------------*)
17494: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17495: begin
17496:   TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )');
17497:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )');
17498:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )');
17499:   SIRegister_TDestinationOptions(CL);
17500:   TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult)');
17501:   TProcOnCopyFileProgressEvent','Procedure(Sender:TObject;FileBytes,
       TransferedBytes:int64;PercentDone:Int64);
17502:   TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String)');
17503:   SIRegister_TcyCopyFiles(CL);
17504:  Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
       ResetAttr:boolean): TCopyFileResult');
17505:  Function cyCopyFileEx( FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
       TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17506:  Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
        ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;'
17507:                      + FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17508: end;
17509:
17510: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17511: begin
17512:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17513:   SIRegister_TcyFileAttributes(CL);
17514:   SIRegister_TSearchRecInstance(CL);
17515:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17516:   TOptions', 'set of TOption');
17517:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17518:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
       Accept:bool;
17519:   TProcOnValidateDirectoryEvent','Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17520:   SIRegister_TcyCustomSearchFiles(CL);
17521:   SIRegister_TcySearchFiles(CL);
17522:  Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17523:  Function IscyFolder( aSRec : TSearchrec) : Boolean');
17524: end;
17525:
17526: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
```

```
17527: begin
17528:  Function jCountChar( const s : string; ch : char) : integer');
17529:  Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings)');
17530:  Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17531:  Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17532:  Function jNormalizeDateTime(const Value:string;theValue:TDateTime;const theFormat:string):string');
17533:  Function jNormalizeDateSeparator( const s : string) : string');
17534:  Function jIsValidDateString( const Value : string) : boolean');
17535:  Function jIsValidTimeString( const Value : string) : boolean');
17536:  Function jIsValidDateTimeString( const Value : string) : boolean');
17537: end;
17538:
17539: procedure SIRegister_kcMapViewer(CL: TPSPascalCompiler);
17540: begin
17541:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapViewer');
17542:   CL.AddTypeS('TMapSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
17543:    +'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
17544:    +', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
17545:    +'EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
17546:    +'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17547:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17548:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17549:   TIntPoint', 'record X : Int64; Y : Int64; end');
17550:   TkcRealPoint', 'record X : Extended; Y : Extended; end');
17551:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17552:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17553:   SIRegister_TCustomDownloadEngine(CL);
17554:   SIRegister_TCustomGeolocationEngine(CL);
17555:   SIRegister_TMapViewer(CL);
17556: end;
17557:
17558: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17559: begin
17560:  (*CL.AddDelphiFunction('Function isFunc( name : TNamePart) : Boolean');
17561:  CL.AddDelphiFunction('Function isUnnamedFunc( name : TNamepart) : Boolean');
17562:  Function isPtrToFunc( name : TNamePart) : Boolean');
17563:  Function isFuncRetFuncPtr( name : TNamePart) : Boolean');
17564:  Function isPtrToFuncRetFuncPtr( name : TNamePart) : Boolean');
17565:  Function GetFuncParam( name : TNamePart) : TNamePart');
17566:  Function isArray( name : TNamePart) : Boolean');
17567:  Function GetArrayPart( name : TNamePart) : TNamePart');
17568:  Function GetIdFromPart( name : TNamePart) : AnsiString');
17569:  Function GetIdPart( name : TNamePart) : TNamePart');
17570:  Function isNamePartPtrToFunc( part : TNamePart) : Boolean');
17571:  Function isAnyBlock( part : TNamePart) : Boolean');*)
17572:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17573:   SIRegister_TLineBreaker(CL);
17574:   CL.AddTypeS('TNameKind', 'Integer');
17575:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17576:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17577:  Function SphericalMod( X : Extended) : Extended');
17578:  Function cSign( Value : Extended) : Extended');
17579:  Function LimitFloat( const eValue, eMin, eMax : Extended) : Extended');
17580:  Function AngleToRadians( iAngle : Extended) : Extended');
17581:  Function RadiansToAngle( eRad : Extended) : Extended');
17582:  Function Cross180( iLong : Double) : Boolean');
17583:  Function Mod180( Value : integer) : Integer');
17584:  Function Mod180Float( Value : Extended) : Extended');
17585:  Function MulDivFloat( a, b, d : Extended) : Extended');
17586:  Function LongDiff( iLong1, iLong2 : Double) : Double');
17587:  Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap)');
17588:  Function Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap');
17589:  Function FixFilePath( const Inpath, CheckPath : string) : string');
17590:  Function UnFixFilePath( const Inpath, CheckPath : string) : string');
17591:  Procedure FillStringList( sl : TStringList; const aText : string)');
17592: end;
17593:
17594:
17595: {A simple Oscilloscope using  TWaveIn class.
17596:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
17597: uses
17598:   Forms,
17599:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
17600:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
17601:   uColorFunctions in 'uColorFunctions.pas',
17602:   AMixer in 'AMixer.pas',
17603:   uSettings in 'uSettings.pas',
17604:   UWavein4 in 'UWavein4.pas',
17605:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
17606:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
17607:
17608:
17609: Functions_max hex in the box maXbox
17610: functionslist.txt
17611: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98
17612:
17613: ********************************************************************************
17614: Procedure
17615: PROCEDURE SIZE 8274 8242 7507 7401 6792 6310 5971 4438 3797 3600
```

```
17616: Procedure **************************Now the Procedure list******************
17617: Procedure ( ACol, ARow : Integer; Items : TStrings)
17618: Procedure ( Agg : TAggregate)
17619: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
17620: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
17621: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
17622: Procedure ( ASender : TObject; const ABytes : Integer)
17623: Procedure ( ASender : TObject; VStream : TStream)
17624: Procedure ( AThread : TIdThread)
17625: Procedure ( AWebModule : TComponent)
17626: Procedure ( Column : TColumn)
17627: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
17628: Procedure ( const iStart : integer; const sText : string)
17629: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
17630: Procedure ( Database : TDatabase; LoginParams : TStrings)
17631: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
17632: Procedure ( DATASET : TDATASET)
17633: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
17634: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
17635: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
17636: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
17637: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
17638: Procedure ( Done : Integer)
17639: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
17640: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
17641: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
17642: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
17643: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
17644: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
17645: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
17646: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
17647: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
17648: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
17649: Procedure ( SENDER : TFIELD; const TEXT : String)
17650: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
17651: Procedure ( Sender : TIdTelnet; const Buffer : String)
17652: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
17653: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
17654: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17655: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
17656: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
17657: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
17658: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
17659: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
17660: Procedure ( Sender : TObject; Button : TMPBtnType)
17661: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
17662: Procedure ( Sender : TObject; Button : TUDBtnType)
17663: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
17664: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
17665: Procedure ( Sender : TObject; Column : TListColumn)
17666: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
17667: Procedure ( Sender : TObject; Connecting : Boolean)
17668: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
17669: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
17670: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
17671: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
17672: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
17673: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
17674: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
17675: Procedure ( Sender : TObject; Index : LongInt)
17676: Procedure ( Sender : TObject; Item : TListItem)
17677: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
17678: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
17679: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
17680: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
17681: Procedure ( Sender : TObject; Item : TListItem; var S : string)
17682: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
17683: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
17684: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
17685: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
17686: Procedure ( Sender : TObject; Node : TTreeNode)
17687: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
17688: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
17689: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
17690: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
17691: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
17692: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
17693: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
17694: Procedure ( Sender : TObject; Rect : TRect)
17695: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
17696: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
17697: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
17698: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
17699: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
17700: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
17701: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
17702: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
```

```
17703: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
17704: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
17705: Procedure ( Sender : TObject; Thread : TServerClientThread)
17706: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
17707: Procedure ( Sender : TObject; Username, Password : string)
17708: Procedure ( Sender : TObject; var AllowChange : Boolean)
17709: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
17710: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
17711: Procedure ( Sender : TObject; var Continue : Boolean)
17712: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
17713: Procedure ( Sender : TObject; var Username : string)
17714: Procedure ( Sender : TObject; Wnd : HWND)
17715: Procedure ( Sender : TToolbar; Button : TToolButton)
17716: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
17717: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
17718: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
17719: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
17720: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
17721: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
17722: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
17723: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
17724: procedure (Sender: TObject)
17725: procedure (Sender: TObject; var Done: Boolean)
17726: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
17727: procedure _T(Name: tbtString; v: Variant);
17728: Procedure AbandonSignalHandler( RtlSigNum : Integer)
17729: Procedure Abort
17730: Procedure About1Click( Sender : TObject)
17731: Procedure Accept( Socket : TSocket)
17732: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
17733: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
17734: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
17735: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
17736: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
17737: Procedure Add( Addend1, Addend2 : TMyBigInt)
17738: Procedure ADD( const AKEY, AVALUE : VARIANT)
17739: Procedure Add( const Key : string; Value : Integer)
17740: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
17741: Procedure ADD( FIELD : TFIELD)
17742: Procedure ADD( ITEM : TMENUITEM)
17743: Procedure ADD( POPUP : TPOPUPMENU)
17744: Procedure AddCharacters( xCharacters : TCharSet)
17745: Procedure AddDriver( const Name : string; List : TStrings)
17746: Procedure AddImages( Value : TCustomImageList)
17747: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
17748: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
17749: Procedure AddLoader( Loader : TBitmapLoader)
17750: Procedure ADDPARAM( VALUE : TPARAM)
17751: Procedure AddPassword( const Password : string)
17752: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
17753: Procedure AddState( oState : TniRegularExpressionState)
17754: Procedure AddStrings( Strings : TStrings);
17755: procedure AddStrings(Strings: TStrings);
17756: Procedure AddStrings1( Strings : TWideStrings);
17757: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
17758: Procedure AddToRecentDocs( const Filename : string)
17759: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
17760: Procedure AllFunctionsList1Click( Sender : TObject)
17761: procedure AllObjectsList1Click(Sender: TObject);
17762: Procedure Allocate( AAllocateBytes : Integer)
17763: procedure AllResourceList1Click(Sender: TObject);
17764: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
17765: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
17766: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
17767: Procedure AnsiFree( var s : AnsiString)
17768: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
17769: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
17770: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
17771: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
17772: Procedure AntiFreeze;
17773: Procedure APPEND
17774: Procedure Append( const S : WideString)
17775: procedure Append(S: string);
17776: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
17777: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
17778: Procedure AppendChunk( Val : OleVariant)
17779: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
17780: Procedure AppendStr( var Dest : string; S : string)
17781: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
17782: Procedure ApplyRange
17783: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17784: Procedure Arrange( Code : TListArrangement)
17785: procedure Assert(expr : Boolean; const msg: string);
17786: procedure Assert2(expr : Boolean; const msg: string);
17787: Procedure Assign( AList : TCustomBucketList)
17788: Procedure Assign( Other : TObject)
17789: Procedure Assign( Source : TDragObject)
17790: Procedure Assign( Source : TPersistent)
17791: Procedure Assign(Source: TPersistent)
```

```
17792: procedure Assign2(mystring, mypath: string);
17793: Procedure AssignCurValues( Source : TDataSet);
17794: Procedure AssignCurValues1( const CurValues : Variant);
17795: Procedure ASSIGNFIELD( FIELD : TFIELD)
17796: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
17797: Procedure AssignFile(var F: Text; FileName: string)
17798: Procedure AssignFile(var F: TextFile; FileName: string)
17799: procedure AssignFileRead(var mystring, myfilename: string);
17800: procedure AssignFileWrite(mystring, myfilename: string);
17801: Procedure AssignTo( Other : TObject)
17802: Procedure AssignValues( Value : TParameters)
17803: Procedure ASSIGNVALUES( VALUE : TPARAMS)
17804: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
17805: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
17806: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
17807: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
17808: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17809: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
17810: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
17811: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
17812: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
17813: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
17814: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
17815: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
17816: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
17817: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17818: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
17819: procedure Beep
17820: Procedure BeepOk
17821: Procedure BeepQuestion
17822: Procedure BeepHand
17823: Procedure BeepExclamation
17824: Procedure BeepAsterisk
17825: Procedure BeepInformation
17826: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
17827: Procedure BeginLayout
17828: Procedure BeginTimer( const Delay, Resolution : Cardinal)
17829: Procedure BeginUpdate
17830: procedure BeginUpdate;
17831: procedure BigScreen1Click(Sender: TObject);
17832: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
17833: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
17834: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
17835: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
17836: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
17837: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
17838: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
17839: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
17840: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
17841: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
17842: Procedure BreakPointMenuClick( Sender : TObject)
17843: procedure BRINGTOFRONT
17844: procedure BringToFront;
17845: Procedure btnBackClick( Sender : TObject)
17846: Procedure btnBrowseClick( Sender : TObject)
17847: Procedure BtnClick( Index : TNavigateBtn)
17848: Procedure btnLargeIconsClick( Sender : TObject)
17849: Procedure BuildAndSendRequest( AURI : TIdURI)
17850: Procedure BuildCache
17851: Procedure BurnMemory( var Buff, BuffLen : integer)
17852: Procedure BurnMemoryStream( Destructo : TMemoryStream)
17853: Procedure CalculateFirstSet
17854: Procedure Cancel
17855: procedure CancelDrag;
17856: Procedure CancelEdit
17857: procedure CANCELHINT
17858: Procedure CancelRange
17859: Procedure CancelUpdates
17860: Procedure CancelWriteBuffer
17861: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
17862: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
17863: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
17864: procedure CaptureScreenFormat(vname: string; vextension: string);
17865: procedure CaptureScreenPNG(vname: string);
17866: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
17867: procedure CASCADE
17868: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
17869: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
17870: Procedure cbPathClick( Sender : TObject)
17871: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
17872: Procedure cedebugAfterExecute( Sender : TPSScript)
17873: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
17874: Procedure cedebugCompile( Sender : TPSScript)
17875: Procedure cedebugExecute( Sender : TPSScript)
17876: Procedure cedebugIdle( Sender : TObject)
17877: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
17878: Procedure CenterHeight( const pc, pcParent : TControl)
17879: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
17880: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
```

```
17881: Procedure Change
17882: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
17883: Procedure Changed
17884: Procedure ChangeDir( const ADirName : string)
17885: Procedure ChangeDirUp
17886: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
17887: Procedure ChangeLevelBy( Value : TChangeRange)
17888: Procedure ChDir(const s: string)
17889: Procedure Check(Status: Integer)
17890: Procedure CheckCommonControl( CC : Integer)
17891: Procedure CHECKFIELDNAME( const FIELDNAME : String)
17892: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
17893: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
17894: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
17895: Procedure CheckToken( T : Char)
17896: procedure CheckToken(t:char)
17897: Procedure CheckTokenSymbol( const S : string)
17898: procedure CheckTokenSymbol(s:string)
17899: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
17900: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17901: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
17902: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
17903: procedure CipherFile1Click(Sender: TObject);
17904: Procedure Clear;
17905: Procedure Clear1Click( Sender : TObject)
17906: Procedure ClearColor( Color : TColor)
17907: Procedure CLEARITEM( AITEM : TMENUITEM)
17908: Procedure ClearMapping
17909: Procedure ClearSelection( KeepPrimary : Boolean)
17910: Procedure ClearWriteBuffer
17911: Procedure Click
17912: Procedure Close
17913: Procedure Close1Click( Sender : TObject)
17914: Procedure CloseDatabase( Database : TDatabase)
17915: Procedure CloseDataSets
17916: Procedure CloseDialog
17917: Procedure CloseFile(var F: Text);
17918: Procedure Closure
17919: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17920: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17921: Procedure CodeCompletionList1Click( Sender : TObject)
17922: Procedure ColEnter
17923: Procedure Collapse
17924: Procedure Collapse( Recurse : Boolean)
17925: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
17926: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
17927: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
17928: Procedure Compile1Click( Sender : TObject)
17929: procedure ComponentCount1Click(Sender: TObject);
17930: Procedure Compress(azipfolder, azipfile: string)
17931: Procedure DeCompress(azipfolder, azipfile: string)
17932: Procedure XZip(azipfolder, azipfile: string)
17933: Procedure XUnZip(azipfolder, azipfile: string)
17934: Procedure Connect( const ATimeout : Integer)
17935: Procedure Connect( Socket : TSocket)
17936: procedure Console1Click(Sender: TObject);
17937: Procedure Continue
17938: Procedure ContinueCount( var Counter : TJclCounter)
17939: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
17940: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
17941: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
17942: Procedure ConvertImage(vsource, vdestination: string);
17943: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
17944: Procedure ConvertBitmap(vsource, vdestination: string);
17945: Procedure ConvertToGray(Cnv: TCanvas);
17946: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
17947: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
17948: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
17949: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
17950: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
17951: Procedure CopyFrom( mbCopy : TMyBigInt)
17952: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
17953: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
17954: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
17955: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
17956: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
17957: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
17958: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
17959: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
17960: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
17961: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17962: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
17963: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17964: Procedure CopyToClipboard
17965: Procedure CountParts
17966: Procedure CreateDataSet
17967: Procedure CreateEmptyFile( const FileName : string)
```

```
17968: Procedure CreateFileFromString( const FileName, Data : string)
17969: Procedure CreateFromDelta( Source : TPacketDataSet)
17970: procedure CREATEHANDLE
17971: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
17972: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
17973: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
17974: Procedure CreateTable
17975: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
17976: procedure CSyntax1Click(Sender: TObject);
17977: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
17978: Procedure CURSORPOSCHANGED
17979: procedure CutFirstDirectory(var S: String)
17980: Procedure DataBaseError(const Message: string)
17981: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
17982: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
17983: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
17984: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
17985: Procedure DBIError(errorCode: Integer)
17986: Procedure DebugOutput( const AText : string)
17987: Procedure DebugRun1Click( Sender : TObject)
17988: procedure Dec;
17989: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
17990: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
17991: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
17992: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
17993: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
17994: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
17995: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
17996: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
17997: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
17998: Procedure Decompile1Click( Sender : TObject)
17999: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
18000: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
18001: Procedure DeferLayout
18002: Procedure defFileread
18003: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
18004: Procedure DelayMicroseconds( const MicroSeconds : Integer)
18005: Procedure Delete
18006: Procedure Delete( const AFilename : string)
18007: Procedure Delete( const Index : Integer)
18008: Procedure DELETE( INDEX : INTEGER)
18009: Procedure Delete( Index : LongInt)
18010: Procedure Delete( Node : TTreeNode)
18011: procedure Delete(var s: AnyString; ifrom, icount: Longint);
18012: Procedure DeleteAlias( const Name : string)
18013: Procedure DeleteDriver( const Name : string)
18014: Procedure DeleteIndex( const Name : string)
18015: Procedure DeleteKey( const Section, Ident : String)
18016: Procedure DeleteRecords
18017: Procedure DeleteRecords( AffectRecords : TAffectRecords)
18018: Procedure DeleteString( var pStr : String; const pDelStr : string)
18019: Procedure DeleteTable
18020: procedure DelphiSite1Click(Sender: TObject);
18021: Procedure Deselect
18022: Procedure Deselect( Node : TTreeNode)
18023: procedure DestroyComponents
18024: Procedure DestroyHandle
18025: Procedure Diff( var X : array of Double)
18026: procedure Diff(var X: array of Double);
18027: Procedure DirCreate( const DirectoryName : String)');
18028: procedure DISABLEALIGN
18029: Procedure DisableConstraints
18030: Procedure Disconnect
18031: Procedure Disconnect( Socket : TSocket)
18032: Procedure Dispose
18033: procedure Dispose(P: PChar)
18034: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
18035: Procedure DoKey( Key : TDBCtrlGridKey)
18036: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18037: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18038: Procedure Dormant
18039: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
18040: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
18041: Procedure DoubleToComp( Value : Double; var Result : Comp)
18042: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18043: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
18044: procedure Draw(X, Y: Integer; Graphic: TGraphic);
18045: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
18046: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18047: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
18048: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18049: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
18050: procedure DrawFocusRect(const Rect: TRect);
18051: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
18052: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18053: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
18054: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
```

```
18055: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18056: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
18057: Procedure DropConnections
18058: Procedure DropDown
18059: Procedure DumpDescription( oStrings : TStrings)
18060: Procedure DumpStateTable( oStrings : TStrings)
18061: Procedure EDIT
18062: Procedure EditButtonClick
18063: Procedure EditFont1Click( Sender : TObject)
18064: procedure Ellipse(X1, Y1, X2, Y2: Integer);
18065: Procedure Ellipse1( const Rect : TRect);
18066: Procedure EMMS
18067: Procedure Encode( ADest : TStream)
18068: procedure ENDDRAG(DROP:BOOLEAN)
18069: Procedure EndEdit( Cancel : Boolean)
18070: Procedure EndTimer
18071: Procedure EndUpdate
18072: Procedure EraseSection( const Section : string)
18073: Procedure Error( const Ident : string)
18074: procedure Error(Ident:Integer)
18075: Procedure ErrorFmt( const Ident : string; const Args : array of const)
18076: Procedure ErrorStr( const Message : string)
18077: procedure ErrorStr(Message:String)
18078: Procedure Exchange( Index1, Index2 : Integer)
18079: procedure Exchange(Index1, Index2: Integer);
18080: Procedure Exec( FileName, Parameters, Directory : string)
18081: Procedure ExecProc
18082: Procedure ExecSQL( UpdateKind : TUpdateKind)
18083: Procedure Execute
18084: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
18085: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
18086: Procedure ExecuteCommand(executeFile, paramstring: string)
18087: Procedure ExecuteShell(executeFile, paramstring: string)
18088: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18089: Procedure ExitThread(ExitCode: Integer); stdcall;
18090: Procedure ExitProcess(ExitCode: Integer); stdcall;
18091: Procedure Expand( AUserName : String; AResults : TStrings)
18092: Procedure Expand( Recurse : Boolean)
18093: Procedure ExportClipboard1Click( Sender : TObject)
18094: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
18095: Procedure ExtractContentFields( Strings : TStrings)
18096: Procedure ExtractCookieFields( Strings : TStrings)
18097: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
18098: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
18099: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
18100: Procedure ExtractQueryFields( Strings : TStrings)
18101: Procedure FastDegToGrad
18102: Procedure FastDegToRad
18103: Procedure FastGradToDeg
18104: Procedure FastGradToRad
18105: Procedure FastRadToDeg
18106: Procedure FastRadToGrad
18107: Procedure FileClose( Handle : Integer)
18108: Procedure FileClose(handle: integer)
18109: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
18110: Procedure FileStructure( AStructure : TIdFTPDataStructure)
18111: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
18112: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
18113: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
18114: Procedure FillChar2(var X: PChar ; count: integer; value: char)
18115: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
18116: Procedure FillIPList
18117: procedure FillRect(const Rect: TRect);
18118: Procedure FillTStrings( AStrings : TStrings)
18119: Procedure FilterOnBookmarks( Bookmarks : array of const)
18120: procedure FinalizePackage(Module: HMODULE)
18121: procedure FindClose;
18122: procedure FindClose2(var F: TSearchRec)
18123: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
18124: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
18125: Procedure FindNearest( const KeyValues : array of const)
18126: Procedure FinishContext
18127: Procedure FIRST
18128: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
18129: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
18130: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
18131: Procedure FlushSchemaCache( const TableName : string)
18132: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
18133: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
18134: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
18135: Procedure FormActivate( Sender : TObject)
18136: procedure FormatLn(const format: String; const args: array of const);  //alias
18137: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18138: Procedure FormCreate( Sender : TObject)
18139: Procedure FormDestroy( Sender : TObject)
18140: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18141: procedure FormOutput1Click(Sender: TObject);
```

```
18142: Procedure FormToHtml( Form : TForm; Path : string)
18143: procedure FrameRect(const Rect: TRect);
18144: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
18145: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
18146: Procedure Free( Buffer : TRecordBuffer)
18147: Procedure Free( Buffer : TValueBuffer)
18148: Procedure Free;
18149: Procedure FreeAndNil(var Obj:TObject)
18150: Procedure FreeImage
18151: procedure FreeMem(P: PChar; Size: Integer)
18152: Procedure FreeTreeData( Tree : TUpdateTree)
18153: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
18154: Procedure FullCollapse
18155: Procedure FullExpand
18156: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
18157: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
18158: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
18159: Procedure Get1( AURL : string; const AResponseContent : TStream);
18160: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
18161: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
18162: Procedure GetAliasNames( List : TStrings)
18163: Procedure GetAliasParams( const AliasName : string; List : TStrings)
18164: Procedure GetApplicationsRunning( Strings : TStrings)
18165: Procedure getBox(aURL, extension: string);
18166: Procedure GetCommandTypes( List : TWideStrings)
18167: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
18168: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
18169: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
18170: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
18171: Procedure GetDatabaseNames( List : TStrings)
18172: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
18173: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
18174: Procedure GetDir(d: byte; var s: string)
18175: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
18176: Procedure GetDriverNames( List : TStrings)
18177: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
18178: Procedure GetDriverParams( const DriverName : string; List : TStrings)
18179: Procedure GetEMails1Click( Sender : TObject)
18180: Procedure getEnvironmentInfo;
18181: Function getEnvironmentString: string;
18182: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
18183: Procedure GetFieldNames( const TableName : string; List : TStrings)
18184: Procedure GetFieldNames( const TableName : string; List : TStrings);
18185: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
18186: Procedure GETFIELDNAMES( LIST : TSTRINGS)
18187: Procedure GetFieldNames1( const TableName : string; List : TStrings);
18188: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
18189: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
18190: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
18191: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
18192: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
18193: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
18194: Procedure GetFormatSettings
18195: Procedure GetFromDIB( var DIB : TBitmapInfo)
18196: Procedure GetFromHDIB( HDIB : HBitmap)
18197: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
18198: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
18199: Procedure GetIcon( Index : Integer; Image : TIcon);
18200: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
18201: Procedure GetIndexInfo( IndexName : string)
18202: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
18203: Procedure GetIndexNames( List : TStrings)
18204: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
18205: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
18206: Procedure GetIndexNames4( const TableName : string; List : TStrings);
18207: Procedure GetInternalResponse
18208: Procedure GETITEMNAMES( LIST : TSTRINGS)
18209: procedure GetMem(P: PChar; Size: Integer)
18210: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
18211: procedure GetPackageDescription(ModuleName: PChar): string)
18212: Procedure GetPackageNames( List : TStrings);
18213: Procedure GetPackageNames1( List : TWideStrings);
18214: Procedure GetParamList( List : TList; const ParamNames : WideString)
18215: Procedure GetProcedureNames( List : TStrings);
18216: Procedure GetProcedureNames( List : TWideStrings);
18217: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
18218: Procedure GetProcedureNames1( List : TStrings);
18219: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
18220: Procedure GetProcedureNames3( List : TWideStrings);
18221: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
18222: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
18223: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
18224: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
18225: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
18226: Procedure GetProviderNames( Names : TWideStrings);
18227: Procedure GetProviderNames( Proc : TGetStrProc)
18228: Procedure GetProviderNames1( Names : TStrings);
18229: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
18230: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
```

```
18231: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
18232: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
18233: Procedure GetSchemaNames( List : TStrings);
18234: Procedure GetSchemaNames1( List : TWideStrings);
18235: Procedure getScriptandRunAsk;
18236: Procedure getScriptandRun(ascript: string);
18237: Procedure getScript(ascript: string);   //alias
18238: Procedure getWebScript(ascript: string);   //alias
18239: Procedure GetSessionNames( List : TStrings)
18240: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
18241: Procedure GetStrings( List : TStrings)
18242: Procedure GetSystemTime; stdcall;
18243: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
18244: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
18245: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
18246: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
18247: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
18248: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
18249: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
18250: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
18251: Procedure GetVisibleWindows( List : Tstrings)
18252: Procedure GoBegin
18253: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
18254: Procedure GotoCurrent( Table : TTable)
18255: procedure GotoEnd1Click(Sender: TObject);
18256: Procedure GotoNearest
18257: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
18258: Procedure HandleException( E : Exception; var Handled : Boolean)
18259: procedure HANDLEMESSAGE
18260: procedure HandleNeeded;
18261: Procedure Head( AURL : string)
18262: Procedure Help( var AHelpContents : TStringList; ACommand : String)
18263: Procedure HexToBinary( Stream : TStream)
18264: procedure HexToBinary(Stream:TStream)
18265: Procedure HideDragImage
18266: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
18267: Procedure HideTraybar
18268: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
18269: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
18270: Procedure HookOSExceptions
18271: Procedure HookSignal( RtlSigNum : Integer)
18272: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
18273: Procedure HTMLSyntax1Click( Sender : TObject)
18274: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
18275: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
18276: Procedure ImportfromClipboard1Click( Sender : TObject)
18277: Procedure ImportfromClipboard2Click( Sender : TObject)
18278: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
18279: procedure Incb(var x: byte);
18280: Procedure Include1Click( Sender : TObject)
18281: Procedure IncludeOFF;  //preprocessing
18282: Procedure IncludeON;
18283: procedure Info1Click(Sender: TObject);
18284: Procedure InitAltRecBuffers( CheckModified : Boolean)
18285: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
18286: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
18287: Procedure InitData( ASource : TDataSet)
18288: Procedure InitDelta( ADelta : TPacketDataSet);
18289: Procedure InitDelta1( const ADelta : OleVariant);
18290: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
18291: Procedure Initialize
18292: procedure InitializePackage(Module: HMODULE)
18293: Procedure INITIATEACTION
18294: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
18295: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
18296: Procedure InitModule( AModule : TComponent)
18297: Procedure InitStdConvs
18298: Procedure InitTreeData( Tree : TUpdateTree)
18299: Procedure INSERT
18300: Procedure Insert( Index : Integer; AClass : TClass)
18301: Procedure Insert( Index : Integer; AComponent : TComponent)
18302: Procedure Insert( Index : Integer; AObject : TObject)
18303: Procedure Insert( Index : Integer; const S : WideString)
18304: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
18305: Procedure Insert(Index: Integer; const S: string);
18306: procedure Insert(Index: Integer; S: string);
18307: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
18308: procedure InsertComponent(AComponent:TComponent)
18309: procedure InsertControl(AControl: TControl);
18310: Procedure InsertIcon( Index : Integer; Image : TIcon)
18311: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
18312: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
18313: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
18314: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
18315: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
18316: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
18317: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
```

```
18318: Procedure InternalBeforeResolve( Tree : TUpdateTree)
18319: Procedure InvalidateModuleCache
18320: Procedure InvalidateTitles
18321: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
18322: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
18323: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
18324: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
18325: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
18326: procedure JavaSyntax1Click(Sender: TObject);
18327: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
18328: Procedure KillDataChannel
18329: Procedure Largefont1Click( Sender : TObject)
18330: Procedure LAST
18331: Procedure LaunchCpl( FileName : string)
18332: Procedure Launch( const AFile : string)
18333: Procedure LaunchFile( const AFile : string)
18334: Procedure LetFileList(FileList: TStringlist; apath: string);
18335: Procedure lineToNumber( xmemo : String; met : boolean)
18336: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
18337: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
18338: Procedure ListViewData( Sender : TObject; Item : TListItem)
18339: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
18340: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
18341: Procedure ListViewDblClick( Sender : TObject)
18342: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18343: Procedure ListDLLExports(const FileName: string; List: TStrings);
18344: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
18345: procedure LoadBytecode1Click(Sender: TObject);
18346: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
18347: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
18348: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
18349: Procedure LoadFromFile( AFileName : string)
18350: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
18351: Procedure LoadFromFile( const FileName : string)
18352: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
18353: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
18354: Procedure LoadFromFile( const FileName : WideString)
18355: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18356: Procedure LoadFromFile(const AFileName: string)
18357: procedure LoadFromFile(FileName: string);
18358: procedure LoadFromFile(FileName:String)
18359: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
18360: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
18361: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
18362: Procedure LoadFromStream( const Stream : TStream)
18363: Procedure LoadFromStream( S : TStream)
18364: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18365: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18366: Procedure LoadFromStream( Stream : TStream)
18367: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
18368: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
18369: procedure LoadFromStream(Stream: TStream);
18370: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
18371: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
18372: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
18373: Procedure LoadLastFile1Click( Sender : TObject)
18374: { LoadIcoToImage loads two icons from resource named NameRes,
18375:   into two image lists ALarge and ASmall}
18376: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
18377: Procedure LoadMemo
18378: Procedure LoadParamsFromIniFile( FFileName : WideString)
18379: Procedure Lock
18380: Procedure Login
18381: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
18382: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
18383: Procedure MakeCaseInsensitive
18384: Procedure MakeDeterministic( var bChanged : boolean)
18385: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
18386: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
18387: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
18388: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
18389: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
18390: Procedure SetRectComplexFormatStr( const S : string)
18391: Procedure SetPolarComplexFormatStr( const S : string)
18392: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
18393: Procedure MakeVisible
18394: Procedure MakeVisible( PartialOK : Boolean)
18395: Procedure Manual1Click( Sender : TObject)
18396: Procedure MarkReachable
18397: Procedure maXbox;  //shows the exe version data in a win box
18398: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
18399: Procedure Memo1Change( Sender : TObject)
18400: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
18401: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
```

```
18402: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18403: procedure Memory1Click(Sender: TObject);
18404: Procedure MERGE( MENU : TMAINMENU)
18405: Procedure MergeChangeLog
18406: procedure MINIMIZE
18407: Procedure MinimizeMaxbox;
18408: Procedure MkDir(const s: string)
18409: Procedure MakeDir(const s: string)');
18410: Procedure ChangeDir(const s: string)');
18411: Function makeFile(const FileName: string): integer)');
18412: Procedure mnuPrintFont1Click( Sender : TObject)
18413: procedure ModalStarted
18414: Procedure Modified
18415: Procedure ModifyAlias( Name : string; List : TStrings)
18416: Procedure ModifyDriver( Name : string; List : TStrings)
18417: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
18418: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
18419: Procedure Move( CurIndex, NewIndex : Integer)
18420: procedure Move(CurIndex, NewIndex: Integer);
18421: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
18422: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
18423: Procedure moveCube( o : TMyLabel)
18424: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
18425: procedure MoveTo(X, Y: Integer);
18426: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
18427: Procedure MovePoint(var x,y:Extended; const angle:Extended);
18428: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
18429: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
18430: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
18431: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
18432: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
18433: procedure New(P: PChar)
18434: procedure New1Click(Sender: TObject);
18435: procedure NewInstance1Click(Sender: TObject);
18436: Procedure NEXT
18437: Procedure NextMonth
18438: Procedure Noop
18439: Procedure NormalizePath( var APath : string)
18440: procedure ObjectBinaryToText(Input, Output: TStream)
18441: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18442: procedure ObjectResourceToText(Input, Output: TStream)
18443: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18444: procedure ObjectTextToBinary(Input, Output: TStream)
18445: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18446: procedure ObjectTextToResource(Input, Output: TStream)
18447: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18448: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
18449: Procedure Open( const UserID : WideString; const Password : WideString);
18450: Procedure Open;
18451: Procedure open1Click( Sender : TObject)
18452: Procedure OpenCdDrive
18453: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
18454: Procedure OpenCurrent
18455: Procedure OpenFile(vfilenamepath: string)
18456: Procedure OpenDirectory1Click( Sender : TObject)
18457: Procedure OpenDir(adir: string);
18458: Procedure OpenIndexFile( const IndexName : string)
18459: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
18460: Procedure OpenWriteBuffer( const AThreshhold : Integer)
18461: Procedure OptimizeMem
18462: Procedure Options1( AURL : string);
18463: Procedure OutputDebugString(lpOutputString : PChar)
18464: Procedure PackBuffer
18465: Procedure Paint
18466: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
18467: Procedure PaintToTBitmap( Target : TBitmap)
18468: Procedure PaletteChanged
18469: Procedure ParentBiDiModeChanged
18470: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
18471: Procedure PasteFromClipboard;
18472: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
18473: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
18474: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
18475: Procedure PError( Text : string)
18476: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18477: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
18478: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
18479: procedure playmp3(mpath: string);
18480: Procedure PlayMP31Click( Sender : TObject)
18481: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
18482: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
18483: procedure PolyBezier(const Points: array of TPoint);
18484: procedure PolyBezierTo(const Points: array of TPoint);
18485: procedure Polygon(const Points: array of TPoint);
18486: procedure Polyline(const Points: array of TPoint);
18487: Procedure Pop
18488: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
18489: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
```

```
18490: Procedure POPUP( X, Y : INTEGER)
18491: Procedure PopupURL(URL : WideString);
18492: Procedure POST
18493: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
18494: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
18495: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
18496: Procedure PostUser( const Email, FirstName, LastName : WideString)
18497: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18498: procedure Pred(X: int64);
18499: Procedure Prepare
18500: Procedure PrepareStatement
18501: Procedure PreProcessXML( AList : TStrings)
18502: Procedure PreventDestruction
18503: Procedure Print( const Caption : string)
18504: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
18505: procedure printf(const format: String; const args: array of const);
18506: Procedure PrintList(Value: TStringList);
18507: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
18508: Procedure Printout1Click( Sender : TObject)
18509: Procedure ProcessHeaders
18510: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
18511: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
18512: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
18513: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
18514: Procedure ProcessMessagesOFF;  //application.processmessages
18515: Procedure ProcessMessagesON;
18516: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
18517: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
18518: Procedure Proclist Size is: 3797 /1415
18519: Procedure procMessClick( Sender : TObject)
18520: Procedure PSScriptCompile( Sender : TPSScript)
18521: Procedure PSScriptExecute( Sender : TPSScript)
18522: Procedure PSScriptLine( Sender : TObject)
18523: Procedure Push( ABoundary : string)
18524: procedure PushItem(AItem: Pointer)
18525: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
18526: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
18527: procedure PutLinuxLines(const Value: string)
18528: Procedure Quit
18529: Procedure RaiseConversionError( const AText : string);
18530: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
18531: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
18532: procedure RaiseException(Ex: TIFException; Param: String);
18533: Procedure RaiseExceptionForLastCmdResult;
18534: procedure RaiseLastException;
18535: procedure RaiseException2;
18536: Procedure RaiseException3(const Msg: string);
18537: Procedure RaiseExcept(const Msg: string);
18538: Procedure RaiseLastOSError
18539: Procedure RaiseLastWin32;
18540: procedure RaiseLastWin32Error)
18541: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
18542: Procedure RandomFillStream( Stream : TMemoryStream)
18543: procedure randomize;
18544: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
18545: Procedure RCS
18546: Procedure Read( Socket : TSocket)
18547: Procedure ReadBlobData
18548: procedure ReadBuffer(Buffer:String;Count:LongInt)
18549: procedure ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
18550: Procedure ReadSection( const Section : string; Strings : TStrings)
18551: Procedure ReadSections( Strings : TStrings)
18552: Procedure ReadSections( Strings : TStrings);
18553: Procedure ReadSections1( const Section : string; Strings : TStrings);
18554: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
18555: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
18556: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
18557: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
18558: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
18559: Procedure Realign;
18560: procedure Rectangle(X1, Y1, X2, Y2: Integer);
18561: Procedure Rectangle1( const Rect : TRect);
18562: Procedure RectCopy( var Dest : TRect; const Source : TRect)
18563: Procedure RectFitToScreen( var R : TRect)
18564: Procedure RectGrow( var R : TRect; const Delta : Integer)
18565: Procedure RectGrowX( var R : TRect; const Delta : Integer)
18566: Procedure RectGrowY( var R : TRect; const Delta : Integer)
18567: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
18568: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
18569: Procedure RectNormalize( var R : TRect)
18570: //  TFileCallbackProcedure = procedure(filename:string);
18571: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
18572: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
18573: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
18574: Procedure Refresh;
18575: Procedure RefreshData( Options : TFetchOptions)
18576: Procedure REFRESHLOOKUPLIST
18577: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
18578: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
```

```
18579: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
18580: Procedure RegisterChanges( Value : TChangeLink)
18581: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
18582: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
18583: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
18584: Procedure ReInitialize( ADelay : Cardinal)
18585: procedure RELEASE
18586: Procedure Remove( const AByteCount : integer)
18587: Procedure REMOVE( FIELD : TFIELD)
18588: Procedure REMOVE( ITEM : TMENUITEM)
18589: Procedure REMOVE( POPUP : TPOPUPMENU)
18590: Procedure RemoveAllPasswords
18591: procedure RemoveComponent(AComponent:TComponent)
18592: Procedure RemoveDir( const ADirName : string)
18593: Procedure RemoveLambdaTransitions( var bChanged : boolean)
18594: Procedure REMOVEPARAM( VALUE : TPARAM)
18595: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
18596: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
18597: Procedure Rename( const ASourceFile, ADestFile : string)
18598: Procedure Rename( const FileName : string; Reload : Boolean)
18599: Procedure RenameTable( const NewTableName : string)
18600: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
18601: Procedure Replace1Click( Sender : TObject)
18602: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
18603: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
18604: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
18605: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
18606: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
18607: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
18608: Procedure Requery( Options : TExecuteOptions)
18609: Procedure Reset
18610: Procedure Reset1Click( Sender : TObject)
18611: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
18612: procedure ResourceExplore1Click(Sender: TObject);
18613: Procedure RestoreContents
18614: Procedure RestoreDefaults
18615: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
18616: Procedure RetrieveHeaders
18617: Procedure RevertRecord
18618: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
18619: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18620: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18621: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
18622: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
18623: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
18624: Procedure RleCompress2( Stream : TStream)
18625: Procedure RleDecompress2( Stream : TStream)
18626: Procedure RmDir(const S: string)
18627: Procedure Rollback
18628: Procedure Rollback( TransDesc : TTransactionDesc)
18629: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
18630: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
18631: Procedure RollbackTrans
18632: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
18633: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
18634: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
18635: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
18636: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
18637: Procedure S_EBox( const AText : string)
18638: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
18639: Procedure S_IBox( const AText : string)
18640: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
18641: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
18642: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
18643: Procedure SampleVarianceAndMean
18644: ( const X : TDynFloatArray; var Variance, Mean : Float)
18645: Procedure Save2Click( Sender : TObject)
18646: Procedure Saveas3Click( Sender : TObject)
18647: Procedure Savebefore1Click( Sender : TObject)
18648: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
18649: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
18650: Procedure SaveConfigFile
18651: Procedure SaveOutput1Click( Sender : TObject)
18652: procedure SaveScreenshotClick(Sender: TObject);
18653: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
18654: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
18655: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
18656: Procedure SaveToFile( AFileName : string)
18657: Procedure SAVETOFILE( const FILENAME : String)
18658: Procedure SaveToFile( const FileName : WideString)
18659: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
18660: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18661: procedure SaveToFile(FileName: string);
18662: procedure SaveToFile(FileName:String)
18663: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
18664: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18665: Procedure SaveToStream( S : TStream)
18666: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
```

```
18667: Procedure SaveToStream( Stream : TStream)
18668: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
18669: procedure SaveToStream(Stream: TStream);
18670: procedure SaveToStream(Stream:TStream)
18671: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
18672: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
18673: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
18674: procedure Say(const sText: string)
18675: Procedure SBytecode1Click( Sender : TObject)
18676: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
18677: procedure ScriptExplorer1Click(Sender: TObject);
18678: Procedure Scroll( Distance : Integer)
18679: Procedure Scroll( DX, DY : Integer)
18680: procedure ScrollBy(DeltaX, DeltaY: Integer);
18681: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
18682: Procedure ScrollTabs( Delta : Integer)
18683: Procedure Search1Click( Sender : TObject)
18684: procedure SearchAndOpenDoc(vfilenamepath: string)
18685: procedure SearchAndOpenFile(vfilenamepath: string)
18686: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
18687: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
18688: Procedure SearchNext1Click( Sender : TObject)
18689: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
18690: Procedure Select1( const Nodes : array of TTreeNode);
18691: Procedure Select2( Nodes : TList);
18692: Procedure SelectNext( Direction : Boolean)
18693: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
18694: Procedure SelfTestPEM  //unit uPSI_cPEM
18695: Procedure Send( AMsg : TIdMessage)
18696: //config forst in const MAILINIFILE = 'maildef.ini';
18697: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
18698: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18699: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18700: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
18701: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
18702: Procedure SendResponse
18703: Procedure SendStream( AStream : TStream)
18704: Procedure Set8087CW( NewCW : Word)
18705: Procedure SetAll( One, Two, Three, Four : Byte)
18706: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
18707: Procedure SetAppDispatcher( const ADispatcher : TComponent)
18708: procedure SetArrayLength;
18709: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
18710: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
18711: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer);  //all init
18712: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
18713: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);');
18714: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);');
18715: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
18716: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
18717: Procedure SetAsHandle( Format : Word; Value : THandle)
18718: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
18719: procedure SetCaptureControl(Control: TControl);
18720: Procedure SetColumnAttributes
18721: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
18722: Procedure SetCustomHeader( const Name, Value : string)
18723: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
18724: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
18725: Procedure SetFocus
18726: procedure SetFocus; virtual;
18727: Procedure SetInitialState
18728: Procedure SetKey
18729: procedure SetLastError(ErrorCode: Integer)
18730: procedure SetLength;
18731: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
18732: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
18733: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
18734: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
18735: Procedure SetParams1( UpdateKind : TUpdateKind);
18736: Procedure SetPassword( const Password : string)
18737: Procedure SetPointer( Ptr : Pointer; Size : Longint)
18738: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
18739: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
18740: Procedure SetProvider( Provider : TComponent)
18741: Procedure SetProxy( const Proxy : string)
18742: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
18743: Procedure SetRange( const StartValues, EndValues : array of const)
18744: Procedure SetRangeEnd
18745: Procedure SetRate( const aPercent, aYear : integer)
18746: procedure SetRate(const aPercent, aYear: integer)
18747: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
18748: Procedure SetSafeCallExceptionMsg( Msg : String)
18749: procedure SETSELTEXTBUF(BUFFER:PCHAR)
18750: Procedure SetSize( AWidth, AHeight : Integer)
18751: procedure SetSize(NewSize:LongInt)
18752: procedure SetString(var s: string; buffer: PChar; len: Integer)
18753: Procedure SetStrings( List : TStrings)
18754: Procedure SetText( Text : PwideChar)
```

```
18755: procedure SetText(Text: PChar);
18756: Procedure SetTextBuf( Buffer : PChar)
18757: procedure SETTEXTBUF(BUFFER:PCHAR)
18758: Procedure SetTick( Value : Integer)
18759: Procedure SetTimeout( ATimeOut : Integer)
18760: Procedure SetTraceEvent( Event : TDBXTraceEvent)
18761: Procedure SetUserName( const UserName : string)
18762: Procedure SetWallpaper( Path : string);
18763: procedure ShellStyle1Click(Sender: TObject);
18764: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
18765: Procedure ShowFileProperties( const FileName : string)
18766: Procedure ShowInclude1Click( Sender : TObject)
18767: Procedure ShowInterfaces1Click( Sender : TObject)
18768: Procedure ShowLastException1Click( Sender : TObject)
18769: Procedure ShowMessage( const Msg : string)
18770: Procedure ShowMessageBig(const aText : string);
18771: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
18772: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
18773: Procedure MsgBig(const aText : string);        //alias
18774: procedure showmessage(mytext: string);
18775: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
18776: procedure ShowMessageFmt(const Msg: string; Params: array of const))
18777: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
18778: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
18779: Procedure ShowSearchDialog( const Directory : string)
18780: Procedure ShowSpecChars1Click( Sender : TObject)
18781: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
18782: Procedure ShredFile( const FileName : string; Times : Integer)
18783: procedure Shuffle(vQ: TStringList);
18784: Procedure ShuffleList( var List : array of Integer; Count : Integer)
18785: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
18786: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
18787: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
18788: Procedure Site( const ACommand : string)
18789: Procedure SkipEOL
18790: Procedure Sleep( ATime : cardinal)
18791: Procedure Sleep( milliseconds : Cardinal)
18792: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
18793: Procedure Slinenumbers1Click( Sender : TObject)
18794: Procedure Sort
18795: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
18796: procedure Speak(const sText: string)   //async like voice
18797: procedure Speak2(const sText: string)  //sync
18798: procedure Split(Str: string;  SubStr: string; List: TStrings);
18799: Procedure SplitNameValue( const Line : string; var Name, Value : string)
18800: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
18801: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
18802: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
18803: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
18804: procedure SQLSyntax1Click(Sender: TObject);
18805: Procedure SRand( Seed : RNG_IntType)
18806: Procedure Start
18807: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
18808: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
18809: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
18810: Procedure StartTransaction( TransDesc : TTransactionDesc)
18811: Procedure Status( var AStatusList : TStringList)
18812: Procedure StatusBar1DblClick( Sender : TObject)
18813: Procedure StepInto1Click( Sender : TObject)
18814: Procedure StepIt
18815: Procedure StepOut1Click( Sender : TObject)
18816: Procedure Stop
18817: procedure stopmp3;
18818: procedure StartWeb(aurl: string);
18819: Procedure Str(aint: integer; astr: string); //of system
18820: Procedure StrDispose( Str : PChar)
18821: procedure StrDispose(Str: PChar)
18822: Procedure StrReplace(var Str: String; Old, New: String);
18823: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
18824: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
18825: Procedure StringToBytes( Value : String; Bytes : array of byte)
18826: procedure StrSet(c : Char; I : Integer; var s : String);
18827: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
18828: Procedure StructureMount( APath : String)
18829: procedure STYLECHANGED(SENDER:TOBJECT)
18830: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
18831: procedure Succ(X: int64);
18832: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
18833: procedure SwapChar(var X,Y: char);  //swapX follows
18834: Procedure SwapFloats( var X, Y : Float)
18835: procedure SwapGrid(grd: TStringGrid);
18836: Procedure SwapOrd( var I, J : Byte);
18837: Procedure SwapOrd( var X, Y : Integer)
18838: Procedure SwapOrd1( var I, J : Shortint);
18839: Procedure SwapOrd2( var I, J : Smallint);
18840: Procedure SwapOrd3( var I, J : Word);
18841: Procedure SwapOrd4( var I, J : Integer);
18842: Procedure SwapOrd5( var I, J : Cardinal);
18843: Procedure SwapOrd6( var I, J : Int64);
```

```
18844: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
18845: Procedure Synchronize1( Method : TMethod);
18846: procedure SyntaxCheck1Click(Sender: TObject);
18847: Procedure SysFreeString(const S: WideString); stdcall;
18848: Procedure TakeOver( Other : TLinearBitmap)
18849: Procedure Talkln(const sText: string)  //async voice
18850: procedure tbtn6resClick(Sender: TObject);
18851: Procedure tbtnUseCaseClick( Sender : TObject)
18852: procedure TerminalStyle1Click(Sender: TObject);
18853: Procedure Terminate
18854: Procedure texSyntax1Click( Sender : TObject)
18855: procedure TextOut(X, Y: Integer; Text: string);
18856: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
18857: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
18858: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
18859: Procedure TextStart
18860: procedure TILE
18861: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
18862: Procedure TitleClick( Column : TColumn)
18863: Procedure ToDo
18864: procedure toolbtnTutorialClick(Sender: TObject);
18865: Procedure Trace1( AURL : string; const AResponseContent : TStream);
18866: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
18867: Procedure Truncate
18868: procedure Tutorial101Click(Sender: TObject);
18869: procedure Tutorial10Statistics1Click(Sender: TObject);
18870: procedure Tutorial11Forms1Click(Sender: TObject);
18871: procedure Tutorial12SQL1Click(Sender: TObject);
18872: Procedure tutorial1Click( Sender : TObject)
18873: Procedure tutorial21Click( Sender : TObject)
18874: Procedure tutorial31Click( Sender : TObject)
18875: Procedure tutorial4Click( Sender : TObject)
18876: Procedure Tutorial5Click( Sender : TObject)
18877: procedure Tutorial6Click(Sender: TObject);
18878: procedure Tutorial91Click(Sender: TObject);
18879: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
18880: procedure UniqueString(var str: AnsiString)
18881: procedure UnloadLoadPackage(Module: HMODULE)
18882: Procedure Unlock
18883: Procedure UNMERGE( MENU : TMAINMENU)
18884: Procedure UnRegisterChanges( Value : TChangeLink)
18885: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
18886: Procedure UnregisterConversionType( const AType : TConvType)
18887: Procedure UnRegisterProvider( Prov : TCustomProvider)
18888: Procedure UPDATE
18889: Procedure UpdateBatch( AffectRecords : TAffectRecords)
18890: Procedure UPDATECURSORPOS
18891: Procedure UpdateFile
18892: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
18893: Procedure UpdateResponse( AResponse : TWebResponse)
18894: Procedure UpdateScrollBar
18895: Procedure UpdateView1Click( Sender : TObject)
18896: procedure Val(const s: string; var n, z: Integer)
18897: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
18898: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
18899: Procedure VariantAdd( const src : Variant; var dst : Variant)
18900: Procedure VariantAnd( const src : Variant; var dst : Variant)
18901: Procedure VariantArrayRedim( var V : Variant; High : Integer)
18902: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
18903: Procedure VariantClear( var V : Variant)
18904: Procedure VariantCpy( const src : Variant; var dst : Variant)
18905: Procedure VariantDiv( const src : Variant; var dst : Variant)
18906: Procedure VariantMod( const src : Variant; var dst : Variant)
18907: Procedure VariantMul( const src : Variant; var dst : Variant)
18908: Procedure VariantOr( const src : Variant; var dst : Variant)
18909: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
18910: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
18911: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
18912: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
18913: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
18914: Procedure VariantShl( const src : Variant; var dst : Variant)
18915: Procedure VariantShr( const src : Variant; var dst : Variant)
18916: Procedure VariantSub( const src : Variant; var dst : Variant)
18917: Procedure VariantXor( const src : Variant; var dst : Variant)
18918: Procedure VarCastError;
18919: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
18920: Procedure VarInvalidOp
18921: Procedure VarInvalidNullOp
18922: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
18923: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
18924: Procedure VarArrayCreateError
18925: Procedure VarResultCheck( AResult : HRESULT);
18926: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
18927: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
18928: Function VarTypeAsText( const AType : TVarType) : string
18929: procedure Voice(const sText: string) //async
18930: procedure Voice2(const sText: string) //sync
18931: Procedure WaitMiliSeconds( AMSec : word)
18932: Procedure WaitMS( AMSec : word)');
```

```
18933: procedure WebCamPic(picname: string);  //eg: c:\mypic.png
18934: Procedure WideAppend( var dst : WideString; const src : WideString)
18935: Procedure WideAssign( var dst : WideString; var src : WideString)
18936: Procedure WideDelete( var dst : WideString; index, count : Integer)
18937: Procedure WideFree( var s : WideString)
18938: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
18939: Procedure WideFromPChar( var dst : WideString; src : PChar)
18940: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
18941: Procedure WideSetLength( var dst : WideString; len : Integer)
18942: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
18943: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
18944: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
18945: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
18946: Procedure HttpGet(const Url: string; Stream:TStream);
18947: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
18948: Procedure WordWrap1Click( Sender : TObject)
18949: Procedure Write( const AOut : string)
18950: Procedure Write( Socket : TSocket)
18951: procedure Write(S: string);
18952: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
18953: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
18954: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
18955: procedure WriteBuffer(Buffer:String;Count:LongInt)
18956: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
18957: Procedure WriteChar( AValue : Char)
18958: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
18959: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
18960: Procedure WriteFloat( const Section, Name : string; Value : Double)
18961: Procedure WriteHeader( AHeader : TStrings)
18962: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
18963: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
18964: Procedure WriteLn( const AOut : string)
18965: procedure Writeln(s: string);
18966: Procedure WriteLog( const FileName, LogLine : string)
18967: Procedure WriteRFCReply( AReply : TIdRFCReply)
18968: Procedure WriteRFCStrings( AStrings : TStrings)
18969: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
18970: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
18971: Procedure WriteString( const Section, Ident, Value : String)
18972: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
18973: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
18974: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
18975: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18976: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18977: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
18978: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
18979: procedure XMLSyntax1Click(Sender: TObject);
18980: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
18981: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
18982: Procedure ZeroFillStream( Stream : TMemoryStream)
18983: procedure XMLSyntax1Click(Sender: TObject);
18984: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
18985: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
18986: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
18987: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
18988: procedure(Sender, Source: TObject;X, Y: Integer)
18989: procedure(Sender, Target: TObject; X, Y: Integer)
18990: procedure(Sender: TObject; ASection, AWidth: Integer)
18991: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
18992: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
18993: procedure(Sender: TObject; var Action: TCloseAction)
18994: procedure(Sender: TObject; var CanClose: Boolean)
18995: procedure(Sender: TObject; var Key: Char);
18996: ProcedureName ProcedureNames ProcedureParametersCursor @
18997:
18998: *************Now Constructors constructor *************
18999:  Size is: 1248 1115 996 628 550 544 501 459 (381)
19000:  Attach( VersionInfoData : Pointer; Size : Integer)
19001:  constructor Create( ABuckets : TBucketListSizes)
19002:  Create( ACallBackWnd : HWND)
19003:  Create( AClient : TCustomTaskDialog)
19004:  Create( AClient : TIdTelnet)
19005:  Create( ACollection : TCollection)
19006:  Create( ACollection : TFavoriteLinkItems)
19007:  Create( ACollection : TTaskDialogButtons)
19008:  Create( AConnection : TIdCustomHTTP)
19009:  Create( ACreateSuspended : Boolean)
19010:  Create( ADataSet : TCustomSQLDataSet)
19011:  CREATE( ADATASET : TDATASET)
19012:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19013:  Create( AGrid : TCustomDBGrid)
19014:  Create( AGrid : TStringGrid; AIndex : Longint)
19015:  Create( AHTTP : TIdCustomHTTP)
19016:  Create( AListItems : TListItems)
19017:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19018:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19019:  Create( AOwner : TCommonCalendar)
19020:  Create( AOwner : TComponent)
19021:  CREATE( AOWNER : TCOMPONENT)
```

```
19022:  Create( AOwner : TCustomListView)
19023:  Create( AOwner : TCustomOutline)
19024:  Create( AOwner : TCustomRichEdit)
19025:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
19026:  Create( AOwner : TCustomTreeView)
19027:  Create( AOwner : TIdUserManager)
19028:  Create( AOwner : TListItems)
19029:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19030:  CREATE( AOWNER : TPERSISTENT)
19031:  Create( AOwner : TPersistent)
19032:  Create( AOwner : TTable)
19033:  Create( AOwner : TTreeNodes)
19034:  Create( AOwner : TWinControl; const ClassName : string)
19035:  Create( AParent : TIdCustomHTTP)
19036:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
19037:  Create( AProvider : TBaseProvider)
19038:  Create( AProvider : TCustomProvider);
19039:  Create( AProvider : TDataSetProvider)
19040:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
19041:  Create( ASocket : TSocket)
19042:  Create( AStrings : TWideStrings)
19043:  Create( AToolBar : TToolBar)
19044:  Create( ATreeNodes : TTreeNodes)
19045:  Create( Autofill : boolean)
19046:  Create( AWebPageInfo : TAbstractWebPageInfo)
19047:  Create( AWebRequest : TWebRequest)
19048:  Create( Collection : TCollection)
19049:  Create( Collection : TIdMessageParts; ABody : TStrings)
19050:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
19051:  Create( Column : TColumn)
19052:  Create( const AConvFamily : TConvFamily; const ADescription : string)
19053:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
19054:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
19055:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
19056:  Create( const ATabSet : TTabSet)
19057:  Create( const Compensate : Boolean)
19058:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
19059:  Create( const FileName : string)
19060:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
19061:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
19062:  Create( const MaskValue : string)
19063:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
19064:  Create( const Prefix : string)
19065:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
19066:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19067:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19068:  Create( CoolBar : TCoolBar)
19069:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
19070:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
19071:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
19072:  Create( DBCtrlGrid : TDBCtrlGrid)
19073:  Create( DSTableProducer : TDSTableProducer)
19074:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
19075:  Create( ErrorCode : DBIResult)
19076:  Create( Field : TBlobField; Mode : TBlobStreamMode)
19077:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
19078:  Create( HeaderControl : TCustomHeaderControl)
19079:  Create( HTTPRequest : TWebRequest)
19080:  Create( iStart : integer; sText : string)
19081:  Create( iValue : Integer)
19082:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
19083:  Create( MciErrNo : MCIERROR; const Msg : string)
19084:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19085:  Create( Message : string; ErrorCode : DBResult)
19086:  Create( Msg : string)
19087:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
19088:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
19089:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
19090:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
19091:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
19092:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
19093:  Create( Owner : TCustomComboBoxEx)
19094:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
19095:  Create( Owner : TPersistent)
19096:  Create( Params : TStrings)
19097:  Create( Size : Cardinal)
19098:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
19099:  Create( StatusBar : TCustomStatusBar)
19100:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
19101:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
19102:  Create(AHandle:Integer)
19103:  Create(AOwner: TComponent); virtual;
19104:  Create(const AURI : string)
19105:  Create(FileName:String;Mode:Word)
19106:  Create(Instance:THandle;ResName:String;ResType:PChar)
```

```
19107:  Create(Stream : TStream)
19108:  Create1( ADataset : TDataset);
19109:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
19110:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19111:  Create2( Other : TObject);
19112:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19113:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19114:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
19115:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
19116:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19117:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
19118:  CreateRes( Ident : Integer);
19119:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
19120:  CreateRes( ResStringRec : PResStringRec);
19121:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
19122:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19123:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19124:  CreateSize( AWidth, AHeight : Integer)
19125:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
19126:
19127: ------------------------------------------------------------------------------
19128: unit uPSI_MathMax;
19129: ------------------------------------------------------------------------------
19130:  CONSTS
19131:   Bernstein: Float = 0.28016949902386913303643649123 07;  // Bernstein constant
19132:   Cbrt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
19133:   Cbrt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
19134:   Cbrt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
19135:   Cbrt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
19136:   CbrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
19137:   Catalan: Float   = 0.91596559417721901505460351493 24;  // Catalan constant
19138:   PiJ:   Float       = 3.14159265358979323846264338327 95;  // PI
19139:   PI:  Extended =  3.14159265358979323846264338327 95);
19140:   PiOn2: Float     = 1.57079632679489661923132169163 98;  // PI / 2
19141:   PiOn3: Float     = 1.04719755119659774615421446109 32;  // PI / 3
19142:   PiOn4: Float     = 0.78539816339744830961566084581 988; // PI / 4
19143:   Sqrt2: Float     = 1.41421356237309504880168872420 97;  // Sqrt(2)
19144:   Sqrt3: Float     = 1.73205080756887729352744634150 59;  // Sqrt(3)
19145:   Sqrt5: Float     = 2.23606797749978969640917366873 13;  // Sqrt(5)
19146:   Sqrt10: Float    = 3.16227766016837933199889354443 27;  // Sqrt(10)
19147:   SqrtPi: Float    = 1.77245385090551602729816748334 11;  // Sqrt(PI)
19148:   Sqrt2Pi: Float   = 2.50662827463100050241576528481 1;   // Sqrt(2 * PI)
19149:   TwoPi: Float     = 6.28318530717958647692528676655 9;   // 2 * PI
19150:   ThreePi: Float   = 9.42477796076937971538793014983 85;  // 3 * PI
19151:   Ln2: Float       = 0.69314718055994530941723212145 818; // Ln(2)
19152:   Ln10: Float      = 2.30258509299404568401799145468 44;  // Ln(10)
19153:   LnPi: Float      = 1.14472988584940017414342735135 31;  // Ln(PI)
19154:   Log2J: Float     = 0.30102999566398119521373889472 449; // Log10(2)
19155:   Log3: Float      = 0.47712125471966243729502790325 512; // Log10(3)
19156:   LogPi: Float     = 0.49714987269413385435126828829 09;  // Log10(PI)
19157:   LogE: Float      = 0.43429448190325182765112891891 661; // Log10(E)
19158:   E: Float         = 2.71828182845904523536028747135 27;  // Natural constant
19159:   hLn2Pi: Float    = 0.91893853320467274178032973640 562; // Ln(2*PI)/2
19160:   inv2Pi: Float    = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
19161:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
19162:   GoldenMean: Float   = 1.61803398874989484820458683436 5638;  // GoldenMean
19163:   EulerMascheroni: Float = 0.57721566490153286060651209008 24;  // Euler GAMMA
19164:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
19165:   StDelta         : Extended = 0.00001;  {delta for difference equations}
19166:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
19167:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
19168:
19169: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
19170: begin
19171:   MetersPerInch = 0.0254; // [1]
19172:   MetersPerFoot = MetersPerInch * 12;
19173:   MetersPerYard = MetersPerFoot * 3;
19174:   MetersPerMile = MetersPerFoot * 5280;
19175:   MetersPerNauticalMiles = 1852;
19176:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
19177:   MetersPerLightSecond = 2.99792458E8; // [5]
19178:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
19179:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
19180:   MetersPerCubit = 0.4572; // [6][7]
19181:   MetersPerFathom = MetersPerFoot * 6;
19182:   MetersPerFurlong = MetersPerYard * 220;
19183:   MetersPerHand = MetersPerInch * 4;
19184:   MetersPerPace = MetersPerInch * 30;
19185:   MetersPerRod = MetersPerFoot * 16.5;
19186:   MetersPerChain = MetersPerRod * 4;
19187:   MetersPerLink = MetersPerChain / 100;
19188:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
19189:   MetersPerPica = MetersPerPoint * 12;
19190:
19191:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
19192:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
19193:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
19194:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
```

```
19195:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
19196:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
19197:
19198:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
19199:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
19200:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
19201:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
19202:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
19203:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
19204:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
19205:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
19206:
19207:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
19208:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
19209:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
19210:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
19211:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
19212:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
19213:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
19214:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
19215:
19216:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
19217:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
19218:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
19219:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
19220:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
19221:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
19222:
19223:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
19224:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
19225:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
19226:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
19227:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
19228:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
19229:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
19230:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
19231:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
19232:
19233:   GramsPerPound = 453.59237; // [1][7]
19234:   GramsPerDrams = GramsPerPound / 256;
19235:   GramsPerGrains = GramsPerPound / 7000;
19236:   GramsPerTons = GramsPerPound * 2000;
19237:   GramsPerLongTons = GramsPerPound * 2240;
19238:   GramsPerOunces = GramsPerPound / 16;
19239:   GramsPerStones = GramsPerPound * 14;
19240:
19241:   MaxAngle 9223372036854775808.0;
19242:   MaxTanH 5678.26170314707197474596553898 54);
19243:   MaxFactorial( 1754);
19244:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
19245:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
19246:   MaxTanH( 354.89135644669199842162284618659);
19247:   MaxFactorial'LongInt'( 170);
19248:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
19249:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
19250:   MaxTanH( 44.36141955583649980270285577332 3);
19251:   MaxFactorial'LongInt'( 33);
19252:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
19253:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
19254:   PiExt( 3.14159265358979323846264338327 95);
19255:   RatioDegToRad( PiExt / 180.0);
19256:   RatioGradToRad( PiExt / 200.0);
19257:   RatioDegToGrad( 200.0 / 180.0);
19258:   RatioGradToDeg( 180.0 / 200.0);
19259:   Crc16PolynomCCITT'LongWord $1021);
19260:  Crc16PolynomIBM'LongWord $8005);
19261:  Crc16Bits'LongInt'( 16);
19262:  Crc16Bytes'LongInt'( 2);
19263:  Crc16HighBit'LongWord $8000);
19264:  NotCrc16HighBit','LongWord $7FFF);
19265:   Crc32PolynomIEEE','LongWord $04C11DB7);
19266:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
19267:  Crc32Koopman','LongWord $741B8CD7);
19268:  Crc32Bits','LongInt'( 32);
19269:  Crc32Bytes','LongInt'( 4);
19270:  Crc32HighBit','LongWord $80000000);
19271:  NotCrc32HighBit','LongWord $7FFFFFFF);
19272:
19273:   MinByte       = Low(Byte);
19274:   MaxByte       = High(Byte);
19275:   MinWord       = Low(Word);
19276:   MaxWord       = High(Word);
19277:   MinShortInt   = Low(ShortInt);
19278:   MaxShortInt   = High(ShortInt);
19279:   MinSmallInt   = Low(SmallInt);
19280:   MaxSmallInt   = High(SmallInt);
19281:   MinLongWord   = LongWord(Low(LongWord));
19282:   MaxLongWord   = LongWord(High(LongWord));
19283:   MinLongInt    = LongInt(Low(LongInt));
```

```
19284:   MaxLongInt    = LongInt(High(LongInt));
19285:   MinInt64      = Int64(Low(Int64));
19286:   MaxInt64      = Int64(High(Int64));
19287:   MinInteger    = Integer(Low(Integer));
19288:   MaxInteger    = Integer(High(Integer));
19289:   MinCardinal   = Cardinal(Low(Cardinal));
19290:   MaxCardinal   = Cardinal(High(Cardinal));
19291:   MinNativeUInt = NativeUInt(Low(NativeUInt));
19292:   MaxNativeUInt = NativeUInt(High(NativeUInt));
19293:   MinNativeInt  = NativeInt(Low(NativeInt));
19294:   MaxNativeInt  = NativeInt(High(NativeInt));
19295:  Function CosH( const Z : Float) : Float;
19296:  Function SinH( const Z : Float) : Float;
19297:  Function TanH( const Z : Float) : Float;
19298:
19299:
19300:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
19301:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
19302:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
19303:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
19304:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
19305:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
19306:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
19307:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
19308:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
19309:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
19310:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
19311:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
19312:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
19313:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
19314:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
19315:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
19316:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
19317:   MaxLog = 709.7827128933840;
19318:   MinLog = -708.3964185322641;
19319:   MaxFac = 170;
19320:   MaxGam = 171.624376956302;
19321:   MaxLgm = 2.556348E+305;
19322:   SingleCompareDelta   = 1.0E-34;
19323:   DoubleCompareDelta   = 1.0E-280;
19324:   {$IFDEF CLR}
19325:   ExtendedCompareDelta = DoubleCompareDelta;
19326:   {$ELSE}
19327:   ExtendedCompareDelta = 1.0E-4400;
19328:   {$ENDIF}
19329:   Bytes1KB  = 1024;
19330:   Bytes1MB  = 1024 * Bytes1KB;
19331:   Bytes1GB  = 1024 * Bytes1MB;
19332:   Bytes64KB = 64 * Bytes1KB;
19333:   Bytes64MB = 64 * Bytes1MB;
19334:   Bytes2GB  = 2 * LongWord(Bytes1GB);
19335:     clBlack32', $FF000000 ));
19336:     clDimGray32', $FF3F3F3F ));
19337:     clGray32', $FF7F7F7F ));
19338:     clLightGray32', $FFBFBFBF ));
19339:     clWhite32', $FFFFFFFF ));
19340:     clMaroon32', $FF7F0000 ));
19341:     clGreen32', $FF007F00 ));
19342:     clOlive32', $FF7F7F00 ));
19343:     clNavy32', $FF00007F ));
19344:     clPurple32', $FF7F007F ));
19345:     clTeal32', $FF007F7F ));
19346:     clRed32', $FFFF0000 ));
19347:     clLime32', $FF00FF00 ));
19348:     clYellow32', $FFFFFF00 ));
19349:     clBlue32', $FF0000FF ));
19350:     clFuchsia32', $FFFF00FF ));
19351:     clAqua32', $FF00FFFF ));
19352:     clAliceBlue32', $FFF0F8FF ));
19353:     clAntiqueWhite32', $FFFAEBD7 ));
19354:     clAquamarine32', $FF7FFFD4 ));
19355:     clAzure32', $FFF0FFFF ));
19356:     clBeige32', $FFF5F5DC ));
19357:     clBisque32', $FFFFE4C4 ));
19358:     clBlancheDalmond32', $FFFFEBCD ));
19359:     clBlueViolet32', $FF8A2BE2 ));
19360:     clBrown32', $FFA52A2A ));
19361:     clBurlyWood32', $FFDEB887 ));
19362:     clCadetblue32', $FF5F9EA0 ));
19363:     clChartReuse32', $FF7FFF00 ));
19364:     clChocolate32', $FFD2691E ));
19365:     clCoral32', $FFFF7F50 ));
19366:     clCornFlowerBlue32', $FF6495ED ));
19367:     clCornSilk32', $FFFFF8DC ));
19368:     clCrimson32', $FFDC143C ));
19369:     clDarkBlue32', $FF00008B ));
19370:     clDarkCyan32', $FF008B8B ));
19371:     clDarkGoldenRod32', $FFB8860B ));
19372:     clDarkGray32', $FFA9A9A9 ));
```

```
19373:     clDarkGreen32', $FF006400 ));
19374:     clDarkGrey32', $FFA9A9A9 ));
19375:     clDarkKhaki32', $FFBDB76B ));
19376:     clDarkMagenta32', $FF8B008B ));
19377:     clDarkOliveGreen32', $FF556B2F ));
19378:     clDarkOrange32', $FFFF8C00 ));
19379:     clDarkOrchid32', $FF9932CC ));
19380:     clDarkRed32', $FF8B0000 ));
19381:     clDarkSalmon32', $FFE9967A ));
19382:     clDarkSeaGreen32', $FF8FBC8F ));
19383:     clDarkSlateBlue32', $FF483D8B ));
19384:     clDarkSlateGray32', $FF2F4F4F ));
19385:     clDarkSlateGrey32', $FF2F4F4F ));
19386:     clDarkTurquoise32', $FF00CED1 ));
19387:     clDarkViolet32', $FF9400D3 ));
19388:     clDeepPink32', $FFFF1493 ));
19389:     clDeepSkyBlue32', $FF00BFFF ));
19390:     clDodgerBlue32', $FF1E90FF ));
19391:     clFireBrick32', $FFB22222 ));
19392:     clFloralWhite32', $FFFFFAF0 ));
19393:     clGainsBoro32', $FFDCDCDC ));
19394:     clGhostWhite32', $FFF8F8FF ));
19395:     clGold32', $FFFFD700 ));
19396:     clGoldenRod32', $FFDAA520 ));
19397:     clGreenYellow32', $FFADFF2F ));
19398:     clGrey32', $FF808080 ));
19399:     clHoneyDew32', $FFF0FFF0 ));
19400:     clHotPink32', $FFFF69B4 ));
19401:     clIndianRed32', $FFCD5C5C ));
19402:     clIndigo32', $FF4B0082 ));
19403:     clIvory32', $FFFFFFF0 ));
19404:     clKhaki32', $FFF0E68C ));
19405:     clLavender32', $FFE6E6FA ));
19406:     clLavenderBlush32', $FFFFF0F5 ));
19407:     clLawnGreen32', $FF7CFC00 ));
19408:     clLemonChiffon32', $FFFFFACD ));
19409:     clLightBlue32', $FFADD8E6 ));
19410:     clLightCoral32', $FFF08080 ));
19411:     clLightCyan32', $FFE0FFFF ));
19412:     clLightGoldenRodYellow32', $FFFAFAD2 ));
19413:     clLightGreen32', $FF90EE90 ));
19414:     clLightGrey32', $FFD3D3D3 ));
19415:     clLightPink32', $FFFFB6C1 ));
19416:     clLightSalmon32', $FFFFA07A ));
19417:     clLightSeagreen32', $FF20B2AA ));
19418:     clLightSkyblue32', $FF87CEFA ));
19419:     clLightSlategray32', $FF778899 ));
19420:     clLightSlategrey32', $FF778899 ));
19421:     clLightSteelblue32', $FFB0C4DE ));
19422:     clLightYellow32', $FFFFFFE0 ));
19423:     clLtGray32', $FFC0C0C0 ));
19424:     clMedGray32', $FFA0A0A4 ));
19425:     clDkGray32', $FF808080 ));
19426:     clMoneyGreen32', $FFC0DCC0 ));
19427:     clLegacySkyBlue32', $FFA6CAF0 ));
19428:     clCream32', $FFFFFBF0 ));
19429:     clLimeGreen32', $FF32CD32 ));
19430:     clLinen32', $FFFAF0E6 ));
19431:     clMediumAquamarine32', $FF66CDAA ));
19432:     clMediumBlue32', $FF0000CD ));
19433:     clMediumOrchid32', $FFBA55D3 ));
19434:     clMediumPurple32', $FF9370DB ));
19435:     clMediumSeaGreen32', $FF3CB371 ));
19436:     clMediumSlateBlue32', $FF7B68EE ));
19437:     clMediumSpringGreen32', $FF00FA9A ));
19438:     clMediumTurquoise32', $FF48D1CC ));
19439:     clMediumVioletRed32', $FFC71585 ));
19440:     clMidnightBlue32', $FF191970 ));
19441:     clMintCream32', $FFF5FFFA ));
19442:     clMistyRose32', $FFFFE4E1 ));
19443:     clMoccasin32', $FFFFE4B5 ));
19444:     clNavajoWhite32', $FFFFDEAD ));
19445:     clOldLace32', $FFFDF5E6 ));
19446:     clOliveDrab32', $FF6B8E23 ));
19447:     clOrange32', $FFFFA500 ));
19448:     clOrangeRed32', $FFFF4500 ));
19449:     clOrchid32', $FFDA70D6 ));
19450:     clPaleGoldenRod32', $FFEEE8AA ));
19451:     clPaleGreen32', $FF98FB98 ));
19452:     clPaleTurquoise32', $FFAFEEEE ));
19453:     clPaleVioletred32', $FFDB7093 ));
19454:     clPapayaWhip32', $FFFFEFD5 ));
19455:     clPeachPuff32', $FFFFDAB9 ));
19456:     clPeru32', $FFCD853F ));
19457:     clPlum32', $FFDDA0DD ));
19458:     clPowderBlue32', $FFB0E0E6 ));
19459:     clRosyBrown32', $FFBC8F8F ));
19460:     clRoyalBlue32', $FF4169E1 ));
19461:     clSaddleBrown32', $FF8B4513 ));
```

```
19462:     clSalmon32', $FFFA8072 ));
19463:     clSandyBrown32', $FFF4A460 ));
19464:     clSeaGreen32', $FF2E8B57 ));
19465:     clSeaShell32', $FFFFF5EE ));
19466:     clSienna32', $FFA0522D ));
19467:     clSilver32', $FFC0C0C0 ));
19468:     clSkyblue32', $FF87CEEB ));
19469:     clSlateBlue32', $FF6A5ACD ));
19470:     clSlateGray32', $FF708090 ));
19471:     clSlateGrey32', $FF708090 ));
19472:     clSnow32', $FFFFFAFA ));
19473:     clSpringgreen32', $FF00FF7F ));
19474:     clSteelblue32', $FF4682B4 ));
19475:     clTan32', $FFD2B48C ));
19476:     clThistle32', $FFD8BFD8 ));
19477:     clTomato32', $FFFF6347 ));
19478:     clTurquoise32', $FF40E0D0 ));
19479:     clViolet32', $FFEE82EE ));
19480:     clWheat32', $FFF5DEB3 ));
19481:     clWhitesmoke32', $FFF5F5F5 ));
19482:     clYellowgreen32', $FF9ACD32 ));
19483:     clTrWhite32', $7FFFFFFF ));
19484:     clTrBlack32', $7F000000 ));
19485:     clTrRed32', $7FFF0000 ));
19486:     clTrGreen32', $7F00FF00 ));
19487:     clTrBlue32', $7F0000FF ));
19488:   // Fixed point math constants
19489:   FixedOne = $10000;  FixedHalf = $7FFF;
19490:   FixedPI  = Round(PI * FixedOne);
19491:   FixedToFloat = 1/FixedOne;
19492:
19493:  Special Types
19494: ************************************************
19495:   type Complex = record
19496:     X, Y : Float;
19497:   end;
19498:   type TVector     = array of Float;
19499:   TIntVector  = array of Integer;
19500:   TCompVector = array of Complex;
19501:   TBoolVector = array of Boolean;
19502:   TStrVector  = array of String;
19503:   TMatrix     = array of TVector;
19504:   TIntMatrix  = array of TIntVector;
19505:   TCompMatrix = array of TCompVector;
19506:   TBoolMatrix = array of TBoolVector;
19507:   TStrMatrix  = array of TStrVector;
19508:   TByteArray = array[0..32767] of byte; !
19509:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
19510:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
19511:   T2StringArray = array of array of string;
19512:   T2IntegerArray = array of array of integer;
19513:   AddTypeS('INT_PTR', 'Integer
19514:   AddTypeS('LONG_PTR', 'Integer
19515:   AddTypeS('UINT_PTR', 'Cardinal
19516:   AddTypeS('ULONG_PTR', 'Cardinal
19517:   AddTypeS('DWORD_PTR', 'ULONG_PTR
19518:   TIntegerDynArray', 'array of Integer
19519:   TCardinalDynArray', 'array of Cardinal
19520:   TWordDynArray', 'array of Word
19521:   TSmallIntDynArray', 'array of SmallInt
19522:   TByteDynArray', 'array of Byte
19523:   TShortIntDynArray', 'array of ShortInt
19524:   TInt64DynArray', 'array of Int64
19525:   TLongWordDynArray', 'array of LongWord
19526:   TSingleDynArray', 'array of Single
19527:   TDoubleDynArray', 'array of Double
19528:   TBooleanDynArray', 'array of Boolean
19529:   TStringDynArray', 'array of string
19530:   TWideStringDynArray', 'array of WideString
19531:   TDynByteArray     = array of Byte;
19532:   TDynShortintArray = array of Shortint;
19533:   TDynSmallintArray = array of Smallint;
19534:   TDynWordArray     = array of Word;
19535:   TDynIntegerArray  = array of Integer;
19536:   TDynLongintArray  = array of Longint;
19537:   TDynCardinalArray = array of Cardinal;
19538:   TDynInt64Array    = array of Int64;
19539:   TDynExtendedArray = array of Extended;
19540:   TDynDoubleArray   = array of Double;
19541:   TDynSingleArray   = array of Single;
19542:   TDynFloatArray    = array of Float;
19543:   TDynPointerArray  = array of Pointer;
19544:   TDynStringArray   = array of string;
19545:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
19546:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
19547:   TSynSearchOptions = set of TSynSearchOption;
19548:
19549:
19550: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
```

```
19551: ------------------------------------------------------------------------------
19552: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
19553: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
19554: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
19555: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19556: function CheckStringSum(vstring: string): integer;
19557: function HexToInt(HexNum: string): LongInt;
19558: function IntToBin(Int: Integer): String;
19559: function BinToInt(Binary: String): Integer;
19560: function HexToBin(HexNum: string): string; external2
19561: function BinToHex(Binary: String): string;
19562: function IntToFloat(i: Integer): double;
19563: function AddThousandSeparator(S: string; myChr: Char): string;
19564: function Max3(const X,Y,Z: Integer): Integer;
19565: procedure Swap(var X,Y: char); // faster without inline
19566: procedure ReverseString(var S: String);
19567: function CharToHexStr(Value: Char): string;
19568: function CharToUniCode(Value: Char): string;
19569: function Hex2Dec(Value: Str002): Byte;
19570: function HexStrCodeToStr(Value: string): string;
19571: function HexToStr(i: integer; value: string): string;
19572: function UniCodeToStr(Value: string): string;
19573: function CRC16(statement: string): string;
19574: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
19575: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
19576: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19577: Procedure ExecuteCommand(executeFile, paramstring: string);
19578: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
19579: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
19580: procedure SearchAndOpenDoc(vfilenamepath: string);
19581: procedure ShowInterfaces(myFile: string);
19582: function Fact2(av: integer): extended;
19583: Function BoolToStr(B: Boolean): string;
19584: Function GCD(x, y : LongInt) : LongInt;
19585: function LCM(m,n: longint): longint;
19586: function GetASCII: string;
19587: function GetItemHeight(Font: TFont): Integer;
19588: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
19589: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
19590: function getHINSTANCE: longword;
19591: function getHMODULE: longword;
19592: function GetASCII: string;
19593: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
19594: function WordIsOk(const AWord: string; var VW: Word): boolean;
19595: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
19596: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
19597: function SafeStr(const s: string): string;
19598: function ExtractUrlPath(const FileName: string): string;
19599: function ExtractUrlName(const FileName: string): string;
19600: function IsInternet: boolean;
19601: function RotateLeft1Bit_u32( Value: uint32): uint32;
19602: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats : Boolean);
19603: procedure getEnvironmentInfo;
19604: procedure AntiFreeze;
19605: function GetCPUSpeed: Double;
19606: function IsVirtualPcGuest : Boolean;
19607: function IsVmWareGuest : Boolean;
19608: procedure StartSerialDialog;
19609: function IsWoW64: boolean;
19610: function IsWow64String(var s: string): Boolean;
19611: procedure StartThreadDemo;
19612: Function RGB(R,G,B: Byte): TColor;
19613: Function Sendln(amess: string): boolean;
19614: Procedure maXbox;
19615: Function AspectRatio(aWidth, aHeight: Integer): String;
19616: function wget(aURL, afile: string): boolean;
19617: procedure PrintList(Value: TStringList);
19618: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
19619: procedure getEnvironmentInfo;
19620: procedure AntiFreeze;
19621: function getBitmap(apath: string): TBitmap;
19622: procedure ShowMessageBig(const aText : string);
19623: function YesNoDialog(const ACaption, AMsg: string): boolean;
19624: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
19625: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19626: //function myStrToBytes(const Value: String): TBytes;
19627: //function myBytesToStr(const Value: TBytes): String;
19628: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19629: function getBitmap(apath: string): TBitmap;
19630: procedure ShowMessageBig(const aText : string);
19631: Function StrToBytes(const Value: String): TBytes;
19632: Function BytesToStr(const Value: TBytes): String;
19633: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19634: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
19635: function FindInPaths(const fileName, paths : String) : String;
19636: procedure initHexArray(var hexn: THexArray);
19637: function josephusG(n,k: integer; var graphout: string): integer;
```

```
19638: function isPowerof2(num: int64): boolean;
19639: function powerOf2(exponent: integer): int64;
19640: function getBigPI: string;
19641: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
19642:  function GetASCIILine: string;
19643: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
19644:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
19645: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19646: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19647: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
19648: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
19649: function isKeypressed: boolean;
19650: function Keypress: boolean;
19651: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19652: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
19653: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
19654: function GetOSName: string;
19655: function GetOSVersion: string;
19656: function GetOSNumber: string;
19657: function getEnvironmentString: string;
19658: procedure StrReplace(var Str: String; Old, New: String);
19659: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19660: function getTeamViewerID: string;
19661: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
19662: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
19663: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19664: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
19665: function StartSocketService: Boolean;
19666: procedure StartSocketServiceForm;
19667: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
19668: function GetFileList1(apath: string): TStringlist;
19669: procedure LetFileList(FileList: TStringlist; apath: string);
19670: procedure StartWeb(aurl: string);
19671: function GetTodayFiles(startdir, amask: string): TStringlist;
19672: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
19673: function JavahashCode(val: string): Integer;
19674: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19675: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
19676: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
19677: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
19678: Procedure ConvertToGray(Cnv: TCanvas);
19679: function GetFileDate(aFile:string; aWithTime:Boolean):string;
19680: procedure ShowMemory;
19681: function ShowMemory2: string;
19682: function getHostIP: string;
19683: procedure ShowBitmap(bmap: TBitmap);
19684: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
19685: function CreateDBGridForm(dblist: TStringList): TListbox;
19686: function isService: boolean;
19687: function isApplication: boolean;
19688: function isTerminalSession: boolean;
19689:
19690:
19691: // News of 3.9.8 up
19692: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19693: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19694: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19695: JvChart - TJvChart Component - 2009 Public
19696: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
19697: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
19698: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
19699: DMath DLL included incl. Demos
19700: Interface Navigator menu/View/Intf Navigator
19701: Unit Explorer menu/Debug/Units Explorer
19702: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
19703: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
19704: Script History to 9 Files WebServer light /Options/Addons/WebServer
19705: Full Text Finder, JVSimLogic Simulator Package
19706: Halt-Stop Program in Menu, WebServer2, Stop Event ,
19707: Conversion Routines, Prebuild Forms, CodeSearch
19708: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19709: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19710: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19711: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
19712: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
19713: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
19714: IDE Reflection API, Session Service Shell S3
19715: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
19716: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
19717: arduino map() function, PMRandom Generator
19718: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
19719: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
19720: REST Test Lib, Multilang Component, Forth Interpreter
19721: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
19722: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
19723: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19724: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19725: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
19726: QRCode Service, add more CFunctions like CDateTime of Synapse
```

```
19727: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
19728: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
19729: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
19730: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
19731: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
19732: BOLD Package, Indy Package5, maTRIx. MATHEMAX
19733: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
19734: emax layers: system-package-component-unit-class-function-block
19735: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
19736: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
19737: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
19738: OpenGL Game Demo: ..Options/Add Ons/Reversi
19739: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
19740: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
19741: 7% performance gain (hot spot profiling)
19742: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
19743: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
19744: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
19745: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
19746:
19747: add routines in 3.9.7.5
19748: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
19749: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
19750: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
19751: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
19752: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
19753: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
19754: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
19755:
19756: ////////////////////////// TestUnits //////////////////////////
19757:   SelftestPEM;
19758:   SelfTestCFundamentUtils;
19759:   SelfTestCFileUtils;
19760:   SelfTestCDateTime;
19761:   SelfTestCTimer;
19762:   SelfTestCRandom;
19763:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
19764:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
19765:
19766:   // Note: There's no need for installing a client certificate in the
19767:   //       webbrowser. The server asks the webbrowser to send a certificate but
19768:   //       if nothing is installed the software will work because the server
19769:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
19770:   //       file: c_cacert.p12  password: c_cakey
19771:
19772:   TGraphicControl = class(TControl)
19773:   private
19774:     FCanvas: TCanvas;
19775:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19776:   protected
19777:     procedure Paint; virtual;
19778:     property Canvas: TCanvas read FCanvas;
19779:   public
19780:     constructor Create(AOwner: TComponent); override;
19781:     destructor Destroy; override;
19782:   end;
19783:
19784:   TCustomControl = class(TWinControl)
19785:   private
19786:     FCanvas: TCanvas;
19787:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19788:   protected
19789:     procedure Paint; virtual;
19790:     procedure PaintWindow(DC: HDC); override;
19791:     property Canvas: TCanvas read FCanvas;
19792:   public
19793:     constructor Create(AOwner: TComponent); override;
19794:     destructor Destroy; override;
19795:   end;
19796:     RegisterPublishedProperties;
19797:     ('ONCHANGE', 'TNotifyEvent', iptrw);
19798:     ('ONCLICK', 'TNotifyEvent', iptrw);
19799:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
19800:     ('ONENTER', 'TNotifyEvent', iptrw);
19801:     ('ONEXIT', 'TNotifyEvent', iptrw);
19802:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
19803:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
19804:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
19805:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
19806:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
19807: //*********************************************************************
19808:   // To stop the while loop, click on Options/Show Include (boolean switch)!
19809:   Control a loop in a script with a form event:
19810:   IncludeON;   //control the while loop
19811:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
19812:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
19813:
19814: //---------------------------------------------------------------------------
19815: //*************mX4 ini-file Configuration***********************************
```

```
19816: //-------------------------------------------------------------------------------
19817:   using config file maxboxdef.ini       menu/Help/Config File
19818:
19819: //*** Definitions for maXbox mX3 ***
19820: [FORM]
19821: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
19822: FONTSIZE=14
19823: EXTENSION=txt
19824: SCREENX=1386
19825: SCREENY=1077
19826: MEMHEIGHT=350
19827: PRINTFONT=Courier New //GUI Settings
19828: LINENUMBERS=Y   //line numbers at gutter in editor at left side
19829: EXCEPTIONLOG=Y  //store excepts + success in 2 log files see below! -menu Debug/Show Last Exceptions
19830: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
19831: BOOTSCRIPT=Y    //enabling load a boot script
19832: MEMORYREPORT=Y  //shows memory report on closing maXbox
19833: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
19834: NAVIGATOR=N     //shows function list at the right side of editor
19835: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
19836: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
19837: [WEB]
19838: IPPORT=8080     //for internal webserver - menu /Options/Add Ons/WebServer
19839: IPHOST=192.168.1.53
19840: ROOTCERT=filepathY
19841: SCERT=filepathY
19842: RSAKEY=filepathY
19843: VERSIONCHECK=Y
19844:
19845: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
19846:
19847: Also possible to set report memory in script to override ini setting
19848: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19849:
19850: After Change the ini file you can reload the file with ../Help/Config Update
19851:
19852: //-------------------------------------------------------------------------------
19853: //**************mX4 maildef.ini ini-file Configuration*********************
19854: //-------------------------------------------------------------------------------
19855: //*** Definitions for maXMail ***
19856: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
19857: [MAXMAIL]
19858: HOST=getmail.softwareschule.ch
19859: USER=mailusername
19860: PASS=password
19861: PORT=110
19862: SSL=Y
19863: BODY=Y
19864: LAST=5
19865:
19866: ADO Connection String:
19867: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
19868:
19869: \452_dbtreeview2access.txt
19870: program TestDbTreeViewMainForm2_ACCESS;
19871:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
19872:                   +Exepath+'\examples\detail.mdb;Persist Security Info=False';
19873:
19874: OpenSSL Lib:  unit ssl_openssl_lib;
19875: {$IFDEF CIL}
19876: const
19877:   {$IFDEF LINUX}
19878:   DLLSSLName = 'libssl.so';
19879:   DLLUtilName = 'libcrypto.so';
19880:   {$ELSE}
19881:   DLLSSLName = 'ssleay32.dll';
19882:   DLLUtilName = 'libeay32.dll';
19883:   {$ENDIF}
19884: {$ELSE}
19885: var
19886:   {$IFNDEF MSWINDOWS}
19887:     {$IFDEF DARWIN}
19888:     DLLSSLName: string = 'libssl.dylib';
19889:     DLLUtilName: string = 'libcrypto.dylib';
19890:     {$ELSE}
19891:     DLLSSLName: string = 'libssl.so';
19892:     DLLUtilName: string = 'libcrypto.so';
19893:     {$ENDIF}
19894:   {$ELSE}
19895:   DLLSSLName: string = 'ssleay32.dll';
19896:   DLLSSLName2: string = 'libssl32.dll';
19897:   DLLUtilName: string = 'libeay32.dll';
19898:   {$ENDIF}
19899: {$ENDIF}
19900:
19901:
19902: //-------------------------------------------------------------------------------
19903: //**************mX4 Macro Tags  ***********************************************
19904: //-------------------------------------------------------- ----------------------
```

```
19905:
19906:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
         E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
19907:
19908: //Tag Macros
19909:
19910:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
19911:
19912: //Tag Macros
19913: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
19914: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
19915: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
19916: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
19917: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
19918: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
19919: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
19920: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
19921: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
19922:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
19923: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
19924: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
19925:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
19926: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
19927:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
19928: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
19929:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
19930:
19931: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
19932:
19933: //Replace Macros
19934:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
19935:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
19936:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
19937:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
19938:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
19939:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
19940:
19941:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
19942:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
19943: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
19944:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
19945:
19946: //-------------------------------------------------------------------------
19947: //**************mX4 ToDo List Tags  ../Help/ToDo List************************
19948: //-------------------------------------------------------- ----------------------
19949:
19950:     while I < sl.Count do begin
19951: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
19952:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
19953:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
19954:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
19955:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
19956:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
19957:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
19958:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
19959:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
19960:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
19961:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
19962:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
19963:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
19964:       Inc(I);
19965:     end;
19966:
19967: //-------------------------------------------------------------------------
19968: //**************mX4 Public  Tools API ****************************************
19969: //-------------------------------------------------------------------------------
19970:   file : unit uPSI_fMain.pas;              {$OTAP} Open Tools API Catalog
19971:  // Those functions concern the editor and preprocessor, all of the IDE
19972:  Example: Call it with maxform1.Info1Click(self)
19973:  Note: Call all Methods with maxForm1., e.g.:
19974:                       maxForm1.ShellStyle1Click(self);
19975:
19976: procedure SIRegister_fMain(CL: TPSPascalCompiler);
19977: begin
19978:  Const('BYTECODE','String 'bytecode.txt'
19979:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
19980:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
19981:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
19982:  Const('PSINC','String PS Includes (*.inc)|*.INC
19983:  Const('DEFFILENAME','String 'firstdemo.txt
19984:  Const('DEFINIFILE','String 'maxboxdef.ini
19985:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
19986:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
19987:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
19988:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
19989:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
19990:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
19991:  Const('INCLUDEBOX','String 'pas_includebox.inc
19992:  Const('BOOTSCRIPT','String 'maxbootscript.txt
```

```
19993:  Const('MBVERSION','String '3.9.9.98
19994:  Const('VERSION','String'3.9.9.98
19995:  Const('MBVER','String '399
19996:  Const('MBVERI','Integer'(399);
19997:  Const('MBVERIALL','Integer'(39998);
19998:  Const('EXENAME','String 'maXbox3.exe
19999:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
20000:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
20001:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
20002:  Const('MXINTERNETCHECK','String 'www.ask.com
20003:  Const('MXMAIL','String 'max@kleiner.com
20004:  Const('TAB','Char #$09);
20005:  Const('CODECOMPLETION','String 'bds_delphi.dci
20006:  SIRegister_TMaxForm1(CL);
20007: end;
20008:
20009:   with FindClass('TForm'),'TMaxForm1') do begin
20010:     memo2', 'TMemo', iptrw);
20011:     memo1', 'TSynMemo', iptrw);
20012:     CB1SCList', 'TComboBox', iptrw);
20013:     mxNavigator', 'TComboBox', iptrw);
20014:     IPHost', 'string', iptrw);
20015:     IPPort', 'integer', iptrw);
20016:     COMPort', 'integer', iptrw);      //3.9.6.4
20017:     Splitter1', 'TSplitter', iptrw);
20018:     PSScript', 'TPSScript', iptrw);
20019:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
20020:     MainMenu1', 'TMainMenu', iptrw);
20021:     Program1', 'TMenuItem', iptrw);
20022:     Compile1', 'TMenuItem', iptrw);
20023:     Files1', 'TMenuItem', iptrw);
20024:     open1', 'TMenuItem', iptrw);
20025:     Save2', 'TMenuItem', iptrw);
20026:     Options1', 'TMenuItem', iptrw);
20027:     Savebefore1', 'TMenuItem', iptrw);
20028:     Largefont1', 'TMenuItem', iptrw);
20029:     sBytecode1', 'TMenuItem', iptrw);
20030:     Saveas3', 'TMenuItem', iptrw);
20031:     Clear1', 'TMenuItem', iptrw);
20032:     Slinenumbers1', 'TMenuItem', iptrw);
20033:     About1', 'TMenuItem', iptrw);
20034:     Search1', 'TMenuItem', iptrw);
20035:     SynPasSyn1', 'TSynPasSyn', iptrw);
20036:     memo1', 'TSynMemo', iptrw);
20037:     SynEditSearch1', 'TSynEditSearch', iptrw);
20038:     WordWrap1', 'TMenuItem', iptrw);
20039:     XPManifest1', 'TXPManifest', iptrw);
20040:     SearchNext1', 'TMenuItem', iptrw);
20041:     Replace1', 'TMenuItem', iptrw);
20042:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
20043:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
20044:     ShowInclude1', 'TMenuItem', iptrw);
20045:     SynEditPrint1', 'TSynEditPrint', iptrw);
20046:     Printout1', 'TMenuItem', iptrw);
20047:     mnPrintColors1', 'TMenuItem', iptrw);
20048:     dlgFilePrint', 'TPrintDialog', iptrw);
20049:     dlgPrintFont1', 'TFontDialog', iptrw);
20050:     mnuPrintFont1', 'TMenuItem', iptrw);
20051:     Include1', 'TMenuItem', iptrw);
20052:     CodeCompletionList1', 'TMenuItem', iptrw);
20053:     IncludeList1', 'TMenuItem', iptrw);
20054:     ImageList1', 'TImageList', iptrw);
20055:     ImageList2', 'TImageList', iptrw);
20056:     CoolBar1', 'TCoolBar', iptrw);
20057:     ToolBar1', 'TToolBar', iptrw);
20058:     tbtnLoad', 'TToolButton', iptrw);
20059:     ToolButton2', 'TToolButton', iptrw);
20060:     tbtnFind', 'TToolButton', iptrw);
20061:     tbtnCompile', 'TToolButton', iptrw);
20062:     tbtnTrans', 'TToolButton', iptrw);
20063:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
20064:     toolbtnTutorial', 'TToolButton', iptrw);
20065:     tbtn6res', 'TToolButton', iptrw);
20066:     ToolButton5', 'TToolButton', iptrw);
20067:     ToolButton1', 'TToolButton', iptrw);
20068:     ToolButton3', 'TToolButton', iptrw);
20069:     statusBar1', 'TStatusBar', iptrw);
20070:     SaveOutput1', 'TMenuItem', iptrw);
20071:     ExportClipboard1', 'TMenuItem', iptrw);
20072:     Close1', 'TMenuItem', iptrw);
20073:     Manual1', 'TMenuItem', iptrw);
20074:     About2', 'TMenuItem', iptrw);
20075:     loadLastfile1', 'TMenuItem', iptrw);
20076:     imglogo', 'TImage', iptrw);
20077:     cedebug', 'TPSScriptDebugger', iptrw);
20078:     debugPopupMenu1', 'TPopupMenu', iptrw);
20079:     BreakPointMenu', 'TMenuItem', iptrw);
20080:     Decompile1', 'TMenuItem', iptrw);
20081:     StepInto1', 'TMenuItem', iptrw);
```

```
20082:     StepOut1', 'TMenuItem', iptrw);
20083:     Reset1', 'TMenuItem', iptrw);
20084:     DebugRun1', 'TMenuItem', iptrw);
20085:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
20086:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20087:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
20088:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20089:     tutorial4', 'TMenuItem', iptrw);
20090:     ExporttoClipboard1', 'TMenuItem', iptrw);
20091:     ImportfromClipboard1', 'TMenuItem', iptrw);
20092:     N4', 'TMenuItem', iptrw);
20093:     N5', 'TMenuItem', iptrw);
20094:     N6', 'TMenuItem', iptrw);
20095:     ImportfromClipboard2', 'TMenuItem', iptrw);
20096:     tutorial1', 'TMenuItem', iptrw);
20097:     N7', 'TMenuItem', iptrw);
20098:     ShowSpecChars1', 'TMenuItem', iptrw);
20099:     OpenDirectory1', 'TMenuItem', iptrw);
20100:     procMess', 'TMenuItem', iptrw);
20101:     tbtnUseCase', 'TToolButton', iptrw);
20102:     ToolButton7', 'TToolButton', iptrw);
20103:     EditFont1', 'TMenuItem', iptrw);
20104:     UseCase1', 'TMenuItem', iptrw);
20105:     tutorial21', 'TMenuItem', iptrw);
20106:     OpenUseCase1', 'TMenuItem', iptrw);
20107:     PSImport_DB1', 'TPSImport_DB', iptrw);
20108:     tutorial31', 'TMenuItem', iptrw);
20109:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
20110:     HTMLSyntax1', 'TMenuItem', iptrw);
20111:     ShowInterfaces1', 'TMenuItem', iptrw);
20112:     Tutorial5', 'TMenuItem', iptrw);
20113:     AllFunctionsList1', 'TMenuItem', iptrw);
20114:     ShowLastException1', 'TMenuItem', iptrw);
20115:     PlayMP31', 'TMenuItem', iptrw);
20116:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
20117:     texSyntax1', 'TMenuItem', iptrw);
20118:     N8', 'TMenuItem', iptrw);
20119:     GetEMails1', 'TMenuItem', iptrw);
20120:     SynCppSyn1', 'TSynCppSyn', iptrw);
20121:     CSyntax1', 'TMenuItem', iptrw);
20122:     Tutorial6', 'TMenuItem', iptrw);
20123:     New1', 'TMenuItem', iptrw);
20124:     AllObjectsList1', 'TMenuItem', iptrw);
20125:     LoadBytecode1', 'TMenuItem', iptrw);
20126:     CipherFile1', 'TMenuItem', iptrw);
20127:     N9', 'TMenuItem', iptrw);
20128:     N10', 'TMenuItem', iptrw);
20129:     Tutorial11', 'TMenuItem', iptrw);
20130:     Tutorial71', 'TMenuItem', iptrw);
20131:     UpdateService1', 'TMenuItem', iptrw);
20132:     PascalSchool1', 'TMenuItem', iptrw);
20133:     Tutorial81', 'TMenuItem', iptrw);
20134:     DelphiSite1', 'TMenuItem', iptrw);
20135:     Output1', 'TMenuItem', iptrw);
20136:     TerminalStyle1', 'TMenuItem', iptrw);
20137:     ReadOnly1', 'TMenuItem', iptrw);
20138:     ShellStyle1', 'TMenuItem', iptrw);
20139:     BigScreen1', 'TMenuItem', iptrw);
20140:     Tutorial91', 'TMenuItem', iptrw);
20141:     SaveOutput2', 'TMenuItem', iptrw);
20142:     N11', 'TMenuItem', iptrw);
20143:     SaveScreenshot', 'TMenuItem', iptrw);
20144:     Tutorial101', 'TMenuItem', iptrw);
20145:     SQLSyntax1', 'TMenuItem', iptrw);
20146:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
20147:     Console1', 'TMenuItem', iptrw);
20148:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
20149:     XMLSyntax1', 'TMenuItem', iptrw);
20150:     ComponentCount1', 'TMenuItem', iptrw);
20151:     NewInstance1', 'TMenuItem', iptrw);
20152:     toolbtnTutorial', 'TToolButton', iptrw);
20153:     Memory1', 'TMenuItem', iptrw);
20154:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
20155:     JavaSyntax1', 'TMenuItem', iptrw);
20156:     SyntaxCheck1', 'TMenuItem', iptrw);
20157:     Tutorial10Statistics1', 'TMenuItem', iptrw);
20158:     ScriptExplorer1', 'TMenuItem', iptrw);
20159:     FormOutput1', 'TMenuItem', iptrw);
20160:     ArduinoDump1', 'TMenuItem', iptrw);
20161:     AndroidDump1', 'TMenuItem', iptrw);
20162:     GotoEnd1', 'TMenuItem', iptrw);
20163:     AllResourceList1', 'TMenuItem', iptrw);
20164:     ToolButton4', 'TToolButton', iptrw);
20165:     tbtn6res', 'TToolButton', iptrw);
20166:     Tutorial11Forms1', 'TMenuItem', iptrw);
20167:     Tutorial12SQL1', 'TMenuItem', iptrw);
20168:     ResourceExplore1', 'TMenuItem', iptrw);
20169:     Info1', 'TMenuItem', iptrw);
20170:     N12', 'TMenuItem', iptrw);
```

```
20171:     CryptoBox1', 'TMenuItem', iptrw);
20172:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
20173:     CipherFile2', 'TMenuItem', iptrw);
20174:     N13', 'TMenuItem', iptrw);
20175:     ModulesCount1', 'TMenuItem', iptrw);
20176:     AddOns2', 'TMenuItem', iptrw);
20177:     N4GewinntGame1', 'TMenuItem', iptrw);
20178:     DocuforAddOns1', 'TMenuItem', iptrw);
20179:     Tutorial14Async1', 'TMenuItem', iptrw);
20180:     Lessons15Review1', 'TMenuItem', iptrw);
20181:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
20182:     PHPSyntax1', 'TMenuItem', iptrw);
20183:     Breakpoint1', 'TMenuItem', iptrw);
20184:     SerialRS2321', 'TMenuItem', iptrw);
20185:     N14', 'TMenuItem', iptrw);
20186:     SynCSSyn1', 'TSynCSSyn', iptrw);
20187:     CSyntax2', 'TMenuItem', iptrw);
20188:     Calculator1', 'TMenuItem', iptrw);
20189:     tbtnSerial', 'TToolButton', iptrw);
20190:     ToolButton8', 'TToolButton', iptrw);
20191:     Tutorial151', 'TMenuItem', iptrw);
20192:     N15', 'TMenuItem', iptrw);
20193:     N16', 'TMenuItem', iptrw);
20194:     ControlBar1', 'TControlBar', iptrw);
20195:     ToolBar2', 'TToolBar', iptrw);
20196:     BtnOpen', 'TToolButton', iptrw);
20197:     BtnSave', 'TToolButton', iptrw);
20198:     BtnPrint', 'TToolButton', iptrw);
20199:     BtnColors', 'TToolButton', iptrw);
20200:     btnClassReport', 'TToolButton', iptrw);
20201:     BtnRotateRight', 'TToolButton', iptrw);
20202:     BtnFullSize', 'TToolButton', iptrw);
20203:     BtnFitToWindowSize', 'TToolButton', iptrw);
20204:     BtnZoomMinus', 'TToolButton', iptrw);
20205:     BtnZoomPlus', 'TToolButton', iptrw);
20206:     Panel1', 'TPanel', iptrw);
20207:     LabelBrettgroesse', 'TLabel', iptrw);
20208:     CB1SCList', 'TComboBox', iptrw);
20209:     ImageListNormal', 'TImageList', iptrw);
20210:     spbtnexplore', 'TSpeedButton', iptrw);
20211:     spbtnexample', 'TSpeedButton', iptrw);
20212:     spbsaveas', 'TSpeedButton', iptrw);
20213:     imglogobox', 'TImage', iptrw);
20214:     EnlargeFont1', 'TMenuItem', iptrw);
20215:     EnlargeFont2', 'TMenuItem', iptrw);
20216:     ShrinkFont1', 'TMenuItem', iptrw);
20217:     ThreadDemo1', 'TMenuItem', iptrw);
20218:     HEXEditor1', 'TMenuItem', iptrw);
20219:     HEXView1', 'TMenuItem', iptrw);
20220:     HEXInspect1', 'TMenuItem', iptrw);
20221:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
20222:     ExporttoHTML1', 'TMenuItem', iptrw);
20223:     ClassCount1', 'TMenuItem', iptrw);
20224:     HTMLOutput1', 'TMenuItem', iptrw);
20225:     HEXEditor2', 'TMenuItem', iptrw);
20226:     Minesweeper1', 'TMenuItem', iptrw);
20227:     N17', 'TMenuItem', iptrw);
20228:     PicturePuzzle1', 'TMenuItem', iptrw);
20229:     sbvclhelp', 'TSpeedButton', iptrw);
20230:     DependencyWalker1', 'TMenuItem', iptrw);
20231:     WebScanner1', 'TMenuItem', iptrw);
20232:     View1', 'TMenuItem', iptrw);
20233:     mnToolbar1', 'TMenuItem', iptrw);
20234:     mnStatusbar2', 'TMenuItem', iptrw);
20235:     mnConsole2', 'TMenuItem', iptrw);
20236:     mnCoolbar2', 'TMenuItem', iptrw);
20237:     mnSplitter2', 'TMenuItem', iptrw);
20238:     WebServer1', 'TMenuItem', iptrw);
20239:     Tutorial17Server1', 'TMenuItem', iptrw);
20240:     Tutorial18Arduino1', 'TMenuItem', iptrw);
20241:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
20242:     PerlSyntax1', 'TMenuItem', iptrw);
20243:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
20244:     PythonSyntax1', 'TMenuItem', iptrw);
20245:     DMathLibrary1', 'TMenuItem', iptrw);
20246:     IntfNavigator1', 'TMenuItem', iptrw);
20247:     EnlargeFontConsole1', 'TMenuItem', iptrw);
20248:     ShrinkFontConsole1', 'TMenuItem', iptrw);
20249:     SetInterfaceList1', 'TMenuItem', iptrw);
20250:     popintfList', 'TPopupMenu', iptrw);
20251:     intfAdd1', 'TMenuItem', iptrw);
20252:     intfDelete1', 'TMenuItem', iptrw);
20253:     intfRefactor1', 'TMenuItem', iptrw);
20254:     Defactor1', 'TMenuItem', iptrw);
20255:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
20256:     Tutorial20Regex', 'TMenuItem', iptrw);
20257:     N18', 'TMenuItem', iptrw);
20258:     ManualE1', 'TMenuItem', iptrw);
20259:     FullTextFinder1', 'TMenuItem', iptrw);
```

```
20260:     Move1', 'TMenuItem', iptrw);
20261:     FractalDemo1', 'TMenuItem', iptrw);
20262:     Tutorial21Android1', 'TMenuItem', iptrw);
20263:     Tutorial0Function1', 'TMenuItem', iptrw);
20264:     SimuLogBox1', 'TMenuItem', iptrw);
20265:     OpenExamples1', 'TMenuItem', iptrw);
20266:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
20267:     JavaScriptSyntax1', 'TMenuItem', iptrw);
20268:     Halt1', 'TMenuItem', iptrw);
20269:     CodeSearch1', 'TMenuItem', iptrw);
20270:     SynRubySyn1', 'TSynRubySyn', iptrw);
20271:     RubySyntax1', 'TMenuItem', iptrw);
20272:     Undo1', 'TMenuItem', iptrw);
20273:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
20274:     LinuxShellScript1', 'TMenuItem', iptrw);
20275:     Rename1', 'TMenuItem', iptrw);
20276:     spdcodesearch', 'TSpeedButton', iptrw);
20277:     Preview1', 'TMenuItem', iptrw);
20278:     Tutorial22Services1', 'TMenuItem', iptrw);
20279:     Tutorial23RealTime1', 'TMenuItem', iptrw);
20280:     Configuration1', 'TMenuItem', iptrw);
20281:     MP3Player1', 'TMenuItem', iptrw);
20282:     DLLSpy1', 'TMenuItem', iptrw);
20283:     SynURIOpener1', 'TSynURIOpener', iptrw);
20284:     SynURISyn1', 'TSynURISyn', iptrw);
20285:     URILinksClicks1', 'TMenuItem', iptrw);
20286:     EditReplace1', 'TMenuItem', iptrw);
20287:     GotoLine1', 'TMenuItem', iptrw);
20288:     ActiveLineColor1', 'TMenuItem', iptrw);
20289:     ConfigFile1', 'TMenuItem', iptrw);
20290:     Sort1Intflist', 'TMenuItem', iptrw);
20291:     Redo1', 'TMenuItem', iptrw);
20292:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
20293:     Tutorial25Configuration1', 'TMenuItem', iptrw);
20294:     IndentSelection1', 'TMenuItem', iptrw);
20295:     UnindentSection1', 'TMenuItem', iptrw);
20296:     SkyStyle1', 'TMenuItem', iptrw);
20297:     N19', 'TMenuItem', iptrw);
20298:     CountWords1', 'TMenuItem', iptrw);
20299:     imbookmarkimages', 'TImageList', iptrw);
20300:     Bookmark11', 'TMenuItem', iptrw);
20301:     N20', 'TMenuItem', iptrw);
20302:     Bookmark21', 'TMenuItem', iptrw);
20303:     Bookmark31', 'TMenuItem', iptrw);
20304:     Bookmark41', 'TMenuItem', iptrw);
20305:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
20306:
20307:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
20308:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
20309:     Procedure PSScriptCompile( Sender : TPSScript)
20310:     Procedure Compile1Click( Sender : TObject)
20311:     Procedure PSScriptExecute( Sender : TPSScript)
20312:     Procedure open1Click( Sender : TObject)
20313:     Procedure Save2Click( Sender : TObject)
20314:     Procedure Savebefore1Click( Sender : TObject)
20315:     Procedure Largefont1Click( Sender : TObject)
20316:     Procedure FormActivate( Sender : TObject)
20317:     Procedure SBytecode1Click( Sender : TObject)
20318:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
20319:     Procedure Saveas3Click( Sender : TObject)
20320:     Procedure Clear1Click( Sender : TObject)
20321:     Procedure Slinenumbers1Click( Sender : TObject)
20322:     Procedure About1Click( Sender : TObject)
20323:     Procedure Search1Click( Sender : TObject)
20324:     Procedure FormCreate( Sender : TObject)
20325:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
20326:                                                 var Action : TSynReplaceAction)
20327:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
20328:     Procedure WordWrap1Click( Sender : TObject)
20329:     Procedure SearchNext1Click( Sender : TObject)
20330:     Procedure Replace1Click( Sender : TObject)
20331:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
20332:     Procedure ShowInclude1Click( Sender : TObject)
20333:     Procedure Printout1Click( Sender : TObject)
20334:     Procedure mnuPrintFont1Click( Sender : TObject)
20335:     Procedure Include1Click( Sender : TObject)
20336:     Procedure FormDestroy( Sender : TObject)
20337:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
20338:     Procedure UpdateView1Click( Sender : TObject)
20339:     Procedure CodeCompletionList1Click( Sender : TObject)
20340:     Procedure SaveOutput1Click( Sender : TObject)
20341:     Procedure ExportClipboard1Click( Sender : TObject)
20342:     Procedure Close1Click( Sender : TObject)
20343:     Procedure Manual1Click( Sender : TObject)
20344:     Procedure LoadLastFile1Click( Sender : TObject)
20345:     Procedure Memo1Change( Sender : TObject)
20346:     Procedure Decompile1Click( Sender : TObject)
20347:     Procedure StepInto1Click( Sender : TObject)
20348:     Procedure StepOut1Click( Sender : TObject)
```

```
20349:     Procedure Reset1Click( Sender : TObject)
20350:     Procedure cedebugAfterExecute( Sender : TPSScript)
20351:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
20352:     Procedure cedebugCompile( Sender : TPSScript)
20353:     Procedure cedebugExecute( Sender : TPSScript)
20354:     Procedure cedebugIdle( Sender : TObject)
20355:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
20356:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
20357:     Procedure BreakPointMenuClick( Sender : TObject)
20358:     Procedure DebugRun1Click( Sender : TObject)
20359:     Procedure tutorial4Click( Sender : TObject)
20360:     Procedure ImportfromClipboard1Click( Sender : TObject)
20361:     Procedure ImportfromClipboard2Click( Sender : TObject)
20362:     Procedure tutorial1Click( Sender : TObject)
20363:     Procedure ShowSpecChars1Click( Sender : TObject)
20364:     Procedure StatusBar1DblClick( Sender : TObject)
20365:     Procedure PSScriptLine( Sender : TObject)
20366:     Procedure OpenDirectory1Click( Sender : TObject)
20367:     Procedure procMessClick( Sender : TObject)
20368:     Procedure tbtnUseCaseClick( Sender : TObject)
20369:     Procedure EditFont1Click( Sender : TObject)
20370:     Procedure tutorial21Click( Sender : TObject)
20371:     Procedure tutorial31Click( Sender : TObject)
20372:     Procedure HTMLSyntax1Click( Sender : TObject)
20373:     Procedure ShowInterfaces1Click( Sender : TObject)
20374:     Procedure Tutorial5Click( Sender : TObject)
20375:     Procedure ShowLastException1Click( Sender : TObject)
20376:     Procedure PlayMP31Click( Sender : TObject)
20377:     Procedure AllFunctionsList1Click( Sender : TObject)
20378:     Procedure texSyntax1Click( Sender : TObject)
20379:     Procedure GetEMails1Click( Sender : TObject)
20380:     procedure DelphiSite1Click(Sender: TObject);
20381:     procedure TerminalStyle1Click(Sender: TObject);
20382:     procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
20383:     procedure ShellStyle1Click(Sender: TObject);
20384:     procedure Console1Click(Sender: TObject);     //3.2
20385:     procedure BigScreen1Click(Sender: TObject);
20386:     procedure Tutorial91Click(Sender: TObject);
20387:     procedure SaveScreenshotClick(Sender: TObject);
20388:     procedure Tutorial101Click(Sender: TObject);
20389:     procedure SQLSyntax1Click(Sender: TObject);
20390:     procedure XMLSyntax1Click(Sender: TObject);
20391:     procedure ComponentCount1Click(Sender: TObject);
20392:     procedure NewInstance1Click(Sender: TObject);
20393:     procedure CSyntax1Click(Sender: TObject);
20394:     procedure Tutorial6Click(Sender: TObject);
20395:     procedure New1Click(Sender: TObject);
20396:     procedure AllObjectsList1Click(Sender: TObject);
20397:     procedure LoadBytecode1Click(Sender: TObject);
20398:     procedure CipherFile1Click(Sender: TObject);  //V3.5
20399:     procedure NewInstance1Click(Sender: TObject);
20400:     procedure toolbtnTutorialClick(Sender: TObject);
20401:     procedure Memory1Click(Sender: TObject);
20402:     procedure JavaSyntax1Click(Sender: TObject);
20403:     procedure SyntaxCheck1Click(Sender: TObject);
20404:     procedure ScriptExplorer1Click(Sender: TObject);
20405:     procedure FormOutput1Click(Sender: TObject);  //V3.6
20406:     procedure GotoEnd1Click(Sender: TObject);
20407:     procedure AllResourceList1Click(Sender: TObject);
20408:     procedure tbtn6resClick(Sender: TObject);   //V3.7
20409:     procedure Info1Click(Sender: TObject);
20410:     procedure Tutorial10Statistics1Click(Sender: TObject);
20411:     procedure Tutorial11Forms1Click(Sender: TObject);
20412:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
20413:     procedure ResourceExplore1Click(Sender: TObject);
20414:     procedure Info1Click(Sender: TObject);
20415:     procedure CryptoBox1Click(Sender: TObject);
20416:     procedure ModulesCount1Click(Sender: TObject);
20417:     procedure N4GewinntGame1Click(Sender: TObject);
20418:     procedure PHPSyntax1Click(Sender: TObject);
20419:     procedure SerialRS2321Click(Sender: TObject);
20420:     procedure CSyntax2Click(Sender: TObject);
20421:     procedure Calculator1Click(Sender: TObject);
20422:     procedure Tutorial13Ciphering1Click(Sender: TObject);
20423:     procedure Tutorial14Async1Click(Sender: TObject);
20424:     procedure PHPSyntax1Click(Sender: TObject);
20425:     procedure BtnZoomPlusClick(Sender: TObject);
20426:     procedure BtnZoomMinusClick(Sender: TObject);
20427:     procedure btnClassReportClick(Sender: TObject);
20428:     procedure ThreadDemo1Click(Sender: TObject);
20429:     procedure HEXView1Click(Sender: TObject);
20430:     procedure ExporttoHTML1Click(Sender: TObject);
20431:     procedure Minesweeper1Click(Sender: TObject);
20432:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
20433:     procedure sbvclhelpClick(Sender: TObject);
20434:     procedure DependencyWalker1Click(Sender: TObject);
20435:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
20436:     procedure WebScanner1Click(Sender: TObject);
20437:     procedure mnToolbar1Click(Sender: TObject);
```

```
20438:     procedure mnStatusbar2Click(Sender: TObject);
20439:     procedure mnConsole2Click(Sender: TObject);
20440:     procedure mnCoolbar2Click(Sender: TObject);
20441:     procedure mnSplitter2Click(Sender: TObject);
20442:     procedure WebServer1Click(Sender: TObject);
20443:     procedure PerlSyntax1Click(Sender: TObject);
20444:     procedure PythonSyntax1Click(Sender: TObject);
20445:     procedure DMathLibrary1Click(Sender: TObject);
20446:     procedure IntfNavigator1Click(Sender: TObject);
20447:     procedure FullTextFinder1Click(Sender: TObject);
20448:     function AppName: string;
20449:     function ScriptName: string;
20450:     function LastName: string;
20451:     procedure FractalDemo1Click(Sender: TObject);
20452:     procedure SimuLogBox1Click(Sender: TObject);
20453:     procedure OpenExamples1Click(Sender: TObject);
20454:     procedure Halt1Click(Sender: TObject);
20455:     procedure Stop;
20456:     procedure CodeSearch1Click(Sender: TObject);
20457:     procedure RubySyntax1Click(Sender: TObject);
20458:     procedure Undo1Click(Sender: TObject);
20459:     procedure LinuxShellScript1Click(Sender: TObject);
20460:     procedure WebScannerDirect(urls: string);
20461:     procedure WebScanner(urls: string);
20462:     procedure LoadInterfaceList2;
20463:     procedure DLLSpy1Click(Sender: TObject);
20464:     procedure Memo1DblClick(Sender: TObject);
20465:     procedure URILinksClicks1Click(Sender: TObject);
20466:     procedure GotoLine1Click(Sender: TObject);
20467:     procedure ConfigFile1Click(Sender: TObject);
20468:     Procedure Sort1IntflistClick( Sender : TObject)
20469:     Procedure Redo1Click( Sender : TObject)
20470:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
20471:     Procedure IndentSelection1Click( Sender : TObject)
20472:     Procedure UnindentSection1Click( Sender : TObject)
20473:     Procedure SkyStyle1Click( Sender : TObject)
20474:     Procedure CountWords1Click( Sender : TObject)
20475:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
20476:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
20477:     Procedure Bookmark11Click( Sender : TObject)
20478:     Procedure Bookmark21Click( Sender : TObject)
20479:     Procedure Bookmark31Click( Sender : TObject)
20480:     Procedure Bookmark41Click( Sender : TObject)
20481:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
20482:     'STATMemoryReport', 'boolean', iptrw);
20483:     'IPPort', 'integer', iptrw);
20484:     'COMPort', 'integer', iptrw);
20485:     'lbintflist', 'TListBox', iptrw);
20486:     Function GetStatChange : boolean
20487:     Procedure SetStatChange( vstat : boolean)
20488:     Function GetActFileName : string
20489:     Procedure SetActFileName( vname : string)
20490:     Function GetLastFileName : string
20491:     Procedure SetLastFileName( vname : string)
20492:     Procedure WebScannerDirect( urls : string)
20493:     Procedure LoadInterfaceList2
20494:     Function GetStatExecuteShell : boolean
20495:     Procedure DoEditorExecuteCommand( EditorCommand : word)
20496:     function GetActiveLineColor: TColor
20497:     procedure SetActiveLineColor(acolor: TColor)
20498:     procedure ScriptListbox1Click(Sender: TObject);
20499:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
20500:     procedure EnlargeGutter1Click(Sender: TObject);
20501:     procedure Tetris1Click(Sender: TObject);
20502:     procedure ToDoList1Click(Sender: TObject);
20503:     procedure ProcessList1Click(Sender: TObject);
20504:     procedure MetricReport1Click(Sender: TObject);
20505:     procedure ProcessList1Click(Sender: TObject);
20506:     procedure TCPSockets1Click(Sender: TObject);
20507:     procedure ConfigUpdate1Click(Sender: TObject);
20508:     procedure ADOWorkbench1Click(Sender: TObject);
20509:     procedure SocketServer1Click(Sender: TObject);
20510:     procedure FormDemo1Click(Sender: TObject);
20511:     procedure Richedit1Click(Sender: TObject);
20512:     procedure SimpleBrowser1Click(Sender: TObject);
20513:     procedure DOSShell1Click(Sender: TObject);
20514:     procedure SynExport1Click(Sender: TObject);
20515:     procedure ExporttoRTF1Click(Sender: TObject);
20516:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
20517:     procedure SOAPTester1Click(Sender: TObject);
20518:     procedure Sniffer1Click(Sender: TObject);
20519:     procedure AutoDetectSyntax1Click(Sender: TObject);
20520:     procedure FPlot1Click(Sender: TObject);
20521:     procedure PasStyle1Click(Sender: TObject);
20522:     procedure Tutorial183RGBLED1Click(Sender: TObject);
20523:     procedure Reversi1Click(Sender: TObject);
20524:     procedure ManualmaXbox1Click(Sender: TObject);
20525:     procedure BlaisePascalMagazine1Click(Sender: TObject);
20526:     procedure AddToDo1Click(Sender: TObject);
```

```
20527:     procedure CreateGUID1Click(Sender: TObject);
20528:     procedure Tutorial27XML1Click(Sender: TObject);
20529:     procedure CreateDLLStub1Click(Sender: TObject);
20530:     procedure Tutorial28DLL1Click(Sender: TObject);');
20531:     procedure ResetKeyPressed;');
20532:     procedure KeyPressedFalse;
20533:     procedure FileChanges1Click(Sender: TObject);');
20534:     procedure OpenGLTry1Click(Sender: TObject);');
20535:     procedure AllUnitList1Click(Sender: TObject);');
20536:     procedure Tutorial29UMLClick(Sender: TObject);
20537:     procedure CreateHeader1Click(Sender: TObject);
20538:     procedure Oscilloscope1Click(Sender: TObject);');
20539:     procedure Tutorial30WOT1Click(Sender: TObject);');
20540:     procedure GetWebScript1Click(Sender: TObject);');
20541:     procedure Checkers1Click(Sender: TObject);');
20542:     procedure TaskMgr1Click(Sender: TObject);');
20543:     procedure WebCam1Click(Sender: TObject);');
20544:
20545:
20546: //-----------------------------------------------------------------------------
20547: //*************mX4 Editor SynEdit Tools API ********************************
20548: //-----------------------------------------------------------------------------
20549: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
20550: begin
20551:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
20552:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
20553:     Constructor Create( AOwner : TComponent)
20554:     SelStart', 'Integer', iptrw);
20555:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
20556:     Procedure UpdateCaret
20557:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
20558:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
20559:     Procedure BeginUndoBlock
20560:     Procedure BeginUpdate
20561:     Function CaretInView : Boolean
20562:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
20563:     Procedure Clear
20564:     Procedure ClearAll
20565:     Procedure ClearBookMark( BookMark : Integer)
20566:     Procedure ClearSelection
20567:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
20568:     Procedure ClearUndo
20569:     Procedure CopyToClipboard
20570:     Procedure CutToClipboard
20571:     Procedure DoCopyToClipboard( const SText : string)
20572:     Procedure EndUndoBlock
20573:     Procedure EndUpdate
20574:     Procedure EnsureCursorPosVisible
20575:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
20576:     Procedure FindMatchingBracket
20577:     Function GetMatchingBracket : TBufferCoord
20578:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
20579:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
20580:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
20581:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
20582:                : TSynHighlighterAttributes) : boolean
20583:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
20584:                var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
20585:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
20586:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
20587:     Procedure GotoBookMark( BookMark : Integer)
20588:     Procedure GotoLineAndCenter( ALine : Integer)
20589:     Function IdentChars : TSynIdentChars
20590:     Procedure InvalidateGutter
20591:     Procedure InvalidateGutterLine( aLine : integer)
20592:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
20593:     Procedure InvalidateLine( Line : integer)
20594:     Procedure InvalidateLines( FirstLine, LastLine : integer)
20595:     Procedure InvalidateSelection
20596:     Function IsBookmark( BookMark : integer) : boolean
20597:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
20598:     Procedure LockUndo
20599:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
20600:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
20601:     Function LineToRow( aLine : integer) : integer
20602:     Function RowToLine( aRow : integer) : integer
20603:     Function NextWordPos : TBufferCoord
20604:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
20605:     Procedure PasteFromClipboard
20606:     Function WordStart : TBufferCoord
20607:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
20608:     Function WordEnd : TBufferCoord
20609:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
20610:     Function PrevWordPos : TBufferCoord
20611:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
20612:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
20613:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
20614:     Procedure Redo
20615:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
```

```
20616:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
20617:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
20618:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
20619:     Procedure SelectAll
20620:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
20621:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
20622:     Procedure SetDefaultKeystrokes
20623:     Procedure SetSelWord
20624:     Procedure SetWordBlock( Value : TBufferCoord)
20625:     Procedure Undo
20626:     Procedure UnlockUndo
20627:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
20628:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
20629:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
20630:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
20631:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
20632:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
20633:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
20634:     Procedure AddFocusControl( aControl : TWinControl)
20635:     Procedure RemoveFocusControl( aControl : TWinControl)
20636:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
20637:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
20638:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
20639:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
20640:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
20641:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
20642:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
20643:     Procedure RemoveLinesPointer
20644:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
20645:     Procedure UnHookTextBuffer
20646:     BlockBegin', 'TBufferCoord', iptrw);
20647:     BlockEnd', 'TBufferCoord', iptrw);
20648:     CanPaste', 'Boolean', iptr);
20649:     CanRedo', 'boolean', iptr);
20650:     CanUndo', 'boolean', iptr);
20651:     CaretX', 'Integer', iptrw);
20652:     CaretY', 'Integer', iptrw);
20653:     CaretXY', 'TBufferCoord', iptrw);
20654:     ActiveLineColor', 'TColor', iptrw);
20655:     DisplayX', 'Integer', iptr);
20656:     DisplayY', 'Integer', iptr);
20657:     DisplayXY', 'TDisplayCoord', iptr);
20658:     DisplayLineCount', 'integer', iptr);
20659:     CharsInWindow', 'Integer', iptr);
20660:     CharWidth', 'integer', iptr);
20661:     Font', 'TFont', iptrw);
20662:     GutterWidth', 'Integer', iptr);
20663:     Highlighter', 'TSynCustomHighlighter', iptrw);
20664:     LeftChar', 'Integer', iptrw);
20665:     LineHeight', 'integer', iptr);
20666:     LinesInWindow', 'Integer', iptr);
20667:     LineText', 'string', iptrw);
20668:     Lines', 'TStrings', iptrw);
20669:     Marks', 'TSynEditMarkList', iptr);
20670:     MaxScrollWidth', 'integer', iptrw);
20671:     Modified', 'Boolean', iptrw);
20672:     PaintLock', 'Integer', iptr);
20673:     ReadOnly', 'Boolean', iptrw);
20674:     SearchEngine', 'TSynEditSearchCustom', iptrw);
20675:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
20676:     SelTabBlock', 'Boolean', iptr);
20677:     SelTabLine', 'Boolean', iptr);
20678:     SelText', 'string', iptrw);
20679:     StateFlags', 'TSynStateFlags', iptr);
20680:     Text', 'string', iptrw);
20681:     TopLine', 'Integer', iptrw);
20682:     WordAtCursor', 'string', iptr);
20683:     WordAtMouse', 'string', iptr);
20684:     UndoList', 'TSynEditUndoList', iptr);
20685:     RedoList', 'TSynEditUndoList', iptr);
20686:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
20687:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
20688:     BorderStyle', 'TSynBorderStyle', iptrw);
20689:     ExtraLineSpacing', 'integer', iptrw);
20690:     Gutter', 'TSynGutter', iptrw);
20691:     HideSelection', 'boolean', iptrw);
20692:     InsertCaret', 'TSynEditCaretType', iptrw);
20693:     InsertMode', 'boolean', iptrw);
20694:     IsScrolling', 'Boolean', iptr);
20695:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
20696:     MaxUndo', 'Integer', iptrw);
20697:     Options', 'TSynEditorOptions', iptrw);
20698:     OverwriteCaret', 'TSynEditCaretType', iptrw);
20699:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
20700:     ScrollHintColor', 'TColor', iptrw);
20701:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
20702:     ScrollBars', 'TScrollStyle', iptrw);
20703:     SelectedColor', 'TSynSelectedColor', iptrw);
20704:     SelectionMode', 'TSynSelectionMode', iptrw);
```

```
20705:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
20706:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
20707:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
20708:     WordWrapGlyph', 'TSynGlyph', iptrw);
20709:     OnChange', 'TNotifyEvent', iptrw);
20710:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
20711:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
20712:     OnContextHelp', 'TContextHelpEvent', iptrw);
20713:     OnDropFiles', 'TDropFilesEvent', iptrw);
20714:     OnGutterClick', 'TGutterClickEvent', iptrw);
20715:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
20716:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
20717:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
20718:     OnPaint', 'TPaintEvent', iptrw);
20719:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
20720:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
20721:     OnReplaceText', 'TReplaceTextEvent', iptrw);
20722:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
20723:     OnStatusChange', 'TStatusChangeEvent', iptrw);
20724:     OnPaintTransient', 'TPaintTransient', iptrw);
20725:     OnScroll', 'TScrollEvent', iptrw);
20726:    end;
20727:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
20728:  Function GetPlaceableHighlighters : TSynHighlighterList
20729:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
20730:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
20731:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
20732:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
20733:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
20734:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
20735:  Function ConvertCodeStringToExtended( AString : String) : String
20736:  Function ConvertExtendedToCodeString( AString : String) : String
20737:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
20738:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
20739:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
20740:
20741:   TSynEditorOption = (
20742:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
20743:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
20744:                                // preceding line
20745:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
20746:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
20747:                                //direction any more
20748:     eoDragDropEditing,         //Allows to select a textblock and drag it in document to another location
20749:     eoDropFiles,               //Allows the editor accept OLE file drops
20750:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
20751:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
20752:     eoGroupUndo,               //When undoing/redoing actions,handle all cont.changes same kind in onecall
20753:                                //instead undoing/redoing each command separately
20754:     eoHalfPageScroll,          //By scrolling with page-up/page-down commands,only scroll half page attime
20755:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
20756:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
20757:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
20758:     eoNoCaret,                 //Makes it so the caret is never visible
20759:     eoNoSelection,             //Disables selecting text
20760:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
20761:     eoScrollByOneLess,         //Forces scrolling to be one less
20762:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
20763:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
20764:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
20765:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
20766:     eoShowSpecialChars,        //Shows the special Characters
20767:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
20768:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
20769:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
20770:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
20771:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
20772:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
20773:
20774:     ********************************Important Editor Short Cuts*****************************);
20775:     Double click to select a word and count words with highlightning.
20776:     Triple click to select a line.
20777:     CTRL+SHIFT+click to extend a selection.
20778:     Drag with the ALT key down to select columns of text !!!
20779:     Drag and drop is supported.
20780:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
20781:     Type CTRL+A to select all.
20782:     Type CTRL+N to set a new line.
20783:     Type CTRL+T to delete a line or token. //Tokenizer
20784:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
20785:     Type CTRL+Shift+T to add ToDo in line and list.
20786:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
20787:     Type CTRL[0..9] to jump or get to bookmarks.
20788:     Type Home to position cursor at beginning of current line and End to position it at end of line.
20789:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
20790:     Page Up and Page Down work as expected.
20791:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
20792:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
20793:
```

```
20794: {$ Short Key Positions Ctrl<A-Z>: }
20795: def
20796:    <A> Select All
20797:    <B> Count Words
20798:    <C> Copy
20799:    <D> Internet Start
20800:    <E> Script List
20801:    <F> Find
20802:    <G> Goto
20803:    <H> Mark Line
20804:    <I> Interface List
20805:    <J> Code Completion
20806:    <K> Console
20807:    <L> Interface List Box
20808:    <M> Font Larger -
20809:    <N> New Line
20810:    <O> Open File
20811:    <P> Font Smaller +
20812:    <Q> Quit
20813:    <R> Replace
20814:    <S> Save!
20815:    <T> Delete Line
20816:    <U> Use Case Editor
20817:    <V> Paste
20818:    <W> URI Links
20819:    <X> Reserved for coding use internal
20820:    <Y> Delete Line
20821:    <Z> Undo
20822:
20823: ref
20824:    F1 Help
20825:    F2 Syntax Check
20826:    F3 Search Next
20827:    F4 New Instance
20828:    F5 Line Mark /Breakpoint
20829:    F6 Goto End
20830:    F7 Debug Step Into
20831:    F8 Debug Step Out
20832:    F9 Compile
20833:    F10 Menu
20834:    F11 Word Count Highlight
20835:    F12 Reserved for coding use internal
20836:
20837:    AddRegisteredVariable( it ,integer');  //for closure!!
20838:    AddRegisteredVariable( sr ,string');  //for closure
20839:    AddRegisteredVariable( bt ,boolean');  //for closure
20840:    AddRegisteredVariable( ft ,double');  //for closure
20841:    AddRegisteredVariable( srlist ,TStringlist');  //for closures
20842:
20843:  def ReservedWords: array[0..82] of string =
20844:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
20845:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
20846:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
20847:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
20848:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
20849:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
20850:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
20851:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
20852:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
20853:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
20854:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
20855:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
20856: //-------------------------------------------------------------------------
20857: //*************End of mX4 Public  Tools API *********************************
20858: //-------------------------------------------------------------------------
20859:
20860: Amount of Functions: 13307
20861: Amount of Procedures: 8291
20862: Amount of Constructors: 1341
20863: Totals of Calls: 22939
20864: SHA1: Win 3.9.9.98 FF1FBC4C54343B301A55089CC01630434175632E
20865:
20866:
20867: ***********************************************************
20868:  Doc Short Manual with 50 Tips!
20869: ***********************************************************
20870: - Install: just save your maxboxdef.ini before and then extract the zip file!
20871: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
20872: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
20873: - Menu: With <Crtl><F3> you can search for code on examples
20874: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
20875: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
20876: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
20877:
20878: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
20879: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
20880: - Context Menu: You can printout your scripts as a pdf-file or html-export
20881: - Context: You do have a context menu with the right mouse click
20882:
```

```
20883: - Menu: With the UseCase Editor you can convert graphic formats too.
20884: - Menu: On menu Options you find Addons as compiled scripts
20885: - IDE: You don't need a mouse to handle maXbox, use shortcuts
20886: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
20887: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
20888: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
20889:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
20890:
20891: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
20892: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
20893: - Code: If you code a loop till key-pressed use function: isKeyPressed;
20894: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
20895: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
20896: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
20897:          to delete and Click and mark to drag a bookmark
20898: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
20899: - IDE: A file info with system and script information you find in menu Program/Information
20900: - IDE: After change the config file in help you can update changes in menu Help/Config Update
20901: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
20902: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
20903: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
20904: - Editor: Set Bookmarks to check your work in app or code
20905: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
20906: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
20907: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
20908:
20909: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
20910: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
20911: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
20912: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
20913: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
20914: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
20915: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
20916: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
20917: - IDE menu /Help/Tools/ open the Task Manager
20918:
20919: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
20920: - Add on when no browser is available start /Options/Add ons/Easy Browser
20921: - Add on SOAP Tester with SOP POST File
20922: - Add on IP Protocol Sniffer with List View
20923: - Add on OpenGL mX Robot Demo for android
20924: - Add on Checkers Game
20925:   Add on Oscilloscope
20926:
20927: - Menu: Help/Tools as a Tool Section with DOS Opener
20928: - Menu Editor: export the code as RTF File
20929: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
20930: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
20931: - Context: Auto Detect of Syntax depending on file extension
20932: - Code: some Windows API function start with w in the name like wGetAtomName();
20933: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
20934: - IDE File Check with menu ..View/File Changes/...
20935: - Context: Create a Header with Create Header in Navigator List at right window
20936: - Code: use  SysErrorMessage to get a real Error Description, Ex.
20937:         RemoveDir('c:\NoSuchFolder') ;
20938:         writeln('System Error Message: '+ SysErrorMessage(GetLastError));
20939: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
20940:
20941: - using DLL example in maXbox: //function: {*****************************************}
20942:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
20943:                                  cb: DWORD): BOOL; //stdcall;;
20944:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
20945:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
20946:      External  'OpenProcess@kernel32.dll stdcall';
20947:
20948: GCC Compile Ex Script
20949: procedure TFormMain_btnCompileClick(Sender: TObject);
20950: begin
20951:   AProcess:= TProcess.Create(Nil);
20952:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
20953:       + ' -o "' + OpenDialog2.FileName + '"';
20954:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
20955:     AProcess.Execute;
20956:     Memo2.Lines.BeginUpdate;
20957:     Memo2.Lines.Clear;
20958:     Memo2.Lines.LoadFromStream(AProcess.Output);
20959:     Memo2.Lines.EndUpdate;
20960:   finally
20961:     AProcess.Free;
20962:   end;
20963: end;
20964:
20965: Stopwatch pattern
20966: Time1:= Time;
20967:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
20968:   if initAndStartBoard then
20969:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
20970:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
20971:   PrintF('%d %s',[Trunc((Time-Time1)*24),
```

```
20972:                   FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
20973:
20974:    POST git-receive-pack (chunked)
20975: Pushing to https://github.com/maxkleiner/maXbox3.git
20976: To https://github.com/maxkleiner/maXbox3.git
20977:    f127d21..c6a98da  masterbox2 -> masterbox2
20978: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
20979:
20980: History Shell Hell
20981: PCT Precompile Technology , mX4 ScriptStudio
20982: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
20983: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
20984: emax layers: system-package-component-unit-class-function-block
20985: new keywords def ref using maXCalcF
20986: UML: use case act class state seq pac comp dep - lib lab
20987: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
20988: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
20989: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
20990: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
20991: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
20992: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
20993: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
20994: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
20995: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
20996: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
20997: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
20998: Inno Install and Setup Routines
20999: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21000: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21001: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21002: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
21003: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
21004: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21005:
21006: Ref:
21007: https://unibe-ch.academia.edu/MaxKleiner
21008: http://www.slideshare.net/maxkleiner1
21009: http://www.scribd.com/max_kleiner
21010: http://www.delphiforfun.org/Programs/Utilities/index.htm
21011: http://www.slideshare.net/maxkleiner1
21012: http://s3.amazonaws.com/PreviewLinks/22959.html
21013: http://www.softwareschule.ch/arduino_training.pdf
21014: http://www.jrsoftware.org/isinfo.php
21015: http://www.be-precision.com/products/precision-builder/express/
21016: http://www.blaisepascal.eu/
21017: http://www.delphibasics.co.uk/
21018: http://www.youtube.com/watch?v=av89HAbqAsI
21019: http://www.angelfire.com/hi5/delphizeus/modal.html
21020: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
21021: http://delphi.org/2014/01/every-android-api-for-delphi/
21022: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
21023: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
21024: =renderBasicSearchNarrative&q=%s';
21025:
21026:
21027:
21028: ************************************************************
21029:  unit List asm internal end
21030: ************************************************************
21031: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
21032: 02 unit SIRegister_IdStrings                   //Indy Sockets
21033: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
21034: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
21035: 05 unit IFSI_WinForm1puzzle;                   //maXbox
21036: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
21037: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
21038: 08 unit RIRegister_MathMax_Routines(exec);    //Jedi & Delphi
21039: 09 unit RIRegister_IdGlobal_Routines(exec);   //Indy Sockets
21040: 10 unit RIRegister_SysUtils_Routines(Exec);   //Delphi
21041: 11 unit uPSI_IdTCPConnection;                 //Indy some functions
21042: 12 unit uPSCompiler.pas;                      //PS kernel functions
21043: 13 unit uPSI_DBCommon;                        //DB Common_Routines and Types
21044: 14 unit uPSI_Printers.pas                     //Delphi VCL
21045: 15 unit uPSI_MPlayer.pas                     //Delphi VCL
21046: 16 unit uPSC_comobj;                         //COM Functions
21047: 17 unit uPSI_Clipbrd;                        //Delphi VCL
21048: 18 unit Filectrl in IFSI_SysUtils_max;       //VCL Runtime
21049: 19 unit uPSI_SqlExpr;                        //DBX3
21050: 20 unit uPSI_ADODB;                          //ADODB
21051: 21 unit uPSI_StrHlpr;                        //String Helper Routines
21052: 22 unit uPSI_DateUtils;                      //Expansion to DateTimeLib
21053: 23 unit uPSI_FileUtils;                      //Expansion to Sys/File Utils
21054: 24 unit JUtils / gsUtils;                    //Jedi / Metabase
21055: 25 unit JvFunctions_max;                     //Jedi Functions
21056: 26 unit HTTPParser;                          //Delphi VCL
21057: 27 unit HTTPUtil;                           //Delphi VCL
21058: 28 unit uPSI_XMLUtil;                       //Delphi VCL
21059: 29 unit uPSI_SOAPHTTPClient;                 //Delphi VCL SOAP WebService V3.5
21060: 30 unit uPSI_Contnrs;                       //Delphi RTL Container of Classes
```

```
21061: 31 unit uPSI_MaskUtils;                    //RTL Edit and Mask functions
21062: 32 unit uPSI_MyBigInt;                     //big integer class with Math
21063: 33 unit uPSI_ConvUtils;                    //Delphi VCL Conversions engine
21064: 34 unit Types_Variants;                    //Delphi\Win32\rtl\sys
21065: 35 unit uPSI_IdHashSHA1;                   //Indy Crypto Lib
21066: 36 unit uPSI_IdHashMessageDigest           //Indy Crypto;
21067: 37 unit uPSI_IdASN1Util;                   //Indy ASN1Utility Routines;
21068: 38 unit uPSI_IdLogFile;                    //Indy Logger from LogBase
21069: 39 unit uPSI_IdIcmpClient;                 //Indy Ping ICMP
21070: 40 unit uPSI_IdHashMessageDigest_max       //Indy Crypto &OpenSSL;
21071: 41 unit uPSI_FileCtrl;                     //Delphi RTL
21072: 42 unit uPSI_Outline;                      //Delphi VCL
21073: 43 unit uPSI_ScktComp;                     //Delphi RTL
21074: 44 unit uPSI_Calendar;                     //Delphi VCL
21075: 45 unit uPSI_VListView                     //VListView;
21076: 46 unit uPSI_DBGrids;                      //Delphi VCL
21077: 47 unit uPSI_DBCtrls;                      //Delphi VCL
21078: 48 unit ide_debugoutput;                   //maXbox
21079: 49 unit uPSI_ComCtrls;                     //Delphi VCL
21080: 50 unit uPSC_stdctrls+;                    //Delphi VCL
21081: 51 unit uPSI_Dialogs;                      //Delphi VCL
21082: 52 unit uPSI_StdConvs;                     //Delphi RTL
21083: 53 unit uPSI_DBClient;                     //Delphi RTL
21084: 54 unit uPSI_DBPlatform;                   //Delphi RTL
21085: 55 unit uPSI_Provider;                     //Delphi RTL
21086: 56 unit uPSI_FMTBcd;                       //Delphi RTL
21087: 57 unit uPSI_DBCGrids;                     //Delphi VCL
21088: 58 unit uPSI_CDSUtil;                      //MIDAS
21089: 59 unit uPSI_VarHlpr;                      //Delphi RTL
21090: 60 unit uPSI_ExtDlgs;                      //Delphi VCL
21091: 61 unit sdpStopwatch;                      //maXbox
21092: 62 unit uPSI_JclStatistics;                //JCL
21093: 63 unit uPSI_JclLogic;                     //JCL
21094: 64 unit uPSI_JclMiscel;                    //JCL
21095: 65 unit uPSI_JclMath_max;                  //JCL RTL
21096: 66 unit uPSI_uTPLb_StreamUtils;            //LockBox 3
21097: 67 unit uPSI_MathUtils;                    //BCB
21098: 68 unit uPSI_JclMultimedia;                //JCL
21099: 69 unit uPSI_WideStrUtils;                 //Delphi API/RTL
21100: 70 unit uPSI_GraphUtil;                    //Delphi RTL
21101: 71 unit uPSI_TypeTrans;                    //Delphi RTL
21102: 72 unit uPSI_HTTPApp;                      //Delphi VCL
21103: 73 unit uPSI_DBWeb;                        //Delphi VCL
21104: 74 unit uPSI_DBBdeWeb;                     //Delphi VCL
21105: 75 unit uPSI_DBXpressWeb;                  //Delphi VCL
21106: 76 unit uPSI_ShadowWnd;                    //Delphi VCL
21107: 77 unit uPSI_ToolWin;                      //Delphi VCL
21108: 78 unit uPSI_Tabs;                         //Delphi VCL
21109: 79 unit uPSI_JclGraphUtils;                //JCL
21110: 80 unit uPSI_JclCounter;                   //JCL
21111: 81 unit uPSI_JclSysInfo;                   //JCL
21112: 82 unit uPSI_JclSecurity;                  //JCL
21113: 83 unit uPSI_JclFileUtils;                 //JCL
21114: 84 unit uPSI_IdUserAccounts;               //Indy
21115: 85 unit uPSI_IdAuthentication;             //Indy
21116: 86 unit uPSI_uTPLb_AES;                    //LockBox 3
21117: 87 unit uPSI_IdHashSHA1;                   //LockBox 3
21118: 88 unit uTPLb_BlockCipher;                 //LockBox 3
21119: 89 unit uPSI_ValEdit.pas;                  //Delphi VCL
21120: 90 unit uPSI_JvVCLUtils;                   //JCL
21121: 91 unit uPSI_JvDBUtil;                     //JCL
21122: 92 unit uPSI_JvDBUtils;                    //JCL
21123: 93 unit uPSI_JvAppUtils;                   //JCL
21124: 94 unit uPSI_JvCtrlUtils;                  //JCL
21125: 95 unit uPSI_JvFormToHtml;                 //JCL
21126: 96 unit uPSI_JvParsing;                    //JCL
21127: 97 unit uPSI_SerDlgs;                      //Toolbox
21128: 98 unit uPSI_Serial;                       //Toolbox
21129: 99 unit uPSI_JvComponent;                  //JCL
21130: 100 unit uPSI_JvCalc;                      //JCL
21131: 101 unit uPSI_JvBdeUtils;                  //JCL
21132: 102 unit uPSI_JvDateUtil;                  //JCL
21133: 103 unit uPSI_JvGenetic;                   //JCL
21134: 104 unit uPSI_JclBase;                     //JCL
21135: 105 unit uPSI_JvUtils;                     //JCL
21136: 106 unit uPSI_JvStrUtil;                   //JCL
21137: 107 unit uPSI_JvStrUtils;                  //JCL
21138: 108 unit uPSI_JvFileUtil;                  //JCL
21139: 109 unit uPSI_JvMemoryInfos;               //JCL
21140: 110 unit uPSI_JvComputerInfo;              //JCL
21141: 111 unit uPSI_JvgCommClasses;              //JCL
21142: 112 unit uPSI_JvgLogics;                   //JCL
21143: 113 unit uPSI_JvLED;                       //JCL
21144: 114 unit uPSI_JvTurtle;                    //JCL
21145: 115 unit uPSI_SortThds; unit uPSI_ThSort;  //maXbox
21146: 116 unit uPSI_JvgUtils;                    //JCL
21147: 117 unit uPSI_JvExprParser;                //JCL
21148: 118 unit uPSI_HexDump;                     //Borland
21149: 119 unit uPSI_DBLogDlg;                    //VCL
```

```
21150: 120 unit uPSI_SqlTimSt;                    //RTL
21151: 121 unit uPSI_JvHtmlParser;                //JCL
21152: 122 unit uPSI_JvgXMLSerializer;           //JCL
21153: 123 unit uPSI_JvJCLUtils;                 //JCL
21154: 124 unit uPSI_JvStrings;                   //JCL
21155: 125 unit uPSI_uTPLb_IntegerUtils;         //TurboPower
21156: 126 unit uPSI_uTPLb_HugeCardinal;         //TurboPower
21157: 127 unit uPSI_uTPLb_HugeCardinalUtils;    //TurboPower
21158: 128 unit uPSI_SynRegExpr;                 //SynEdit
21159: 129 unit uPSI_StUtils;                    //SysTools4
21160: 130 unit uPSI_StToHTML;                   //SysTools4
21161: 131 unit uPSI_StStrms;                    //SysTools4
21162: 132 unit uPSI_StFIN;                      //SysTools4
21163: 133 unit uPSI_StAstroP;                   //SysTools4
21164: 134 unit uPSI_StStat;                     //SysTools4
21165: 135 unit uPSI_StNetCon;                   //SysTools4
21166: 136 unit uPSI_StDecMth;                   //SysTools4
21167: 137 unit uPSI_StOStr;                     //SysTools4
21168: 138 unit uPSI_StPtrns;                    //SysTools4
21169: 139 unit uPSI_StNetMsg;                   //SysTools4
21170: 140 unit uPSI_StMath;                     //SysTools4
21171: 141 unit uPSI_StExpEng;                   //SysTools4
21172: 142 unit uPSI_StCRC;                      //SysTools4
21173: 143 unit uPSI_StExport,                   //SysTools4
21174: 144 unit uPSI_StExpLog,                   //SysTools4
21175: 145 unit uPSI_ActnList;                   //Delphi VCL
21176: 146 unit uPSI_jpeg;                       //Borland
21177: 147 unit uPSI_StRandom;                   //SysTools4
21178: 148 unit uPSI_StDict;                     //SysTools4
21179: 149 unit uPSI_StBCD;                      //SysTools4
21180: 150 unit uPSI_StTxtDat;                   //SysTools4
21181: 151 unit uPSI_StRegEx;                    //SysTools4
21182: 152 unit uPSI_IMouse;                     //VCL
21183: 153 unit uPSI_SyncObjs;                   //VCL
21184: 154 unit uPSI_AsyncCalls;                 //Hausladen
21185: 155 unit uPSI_ParallelJobs;               //Saraiva
21186: 156 unit uPSI_Variants;                   //VCL
21187: 157 unit uPSI_VarCmplx;                   //VCL Wolfram
21188: 158 unit uPSI_DTDSchema;                  //VCL
21189: 159 unit uPSI_ShLwApi;                    //Brakel
21190: 160 unit uPSI_IBUtils;                    //VCL
21191: 161 unit uPSI_CheckLst;                   //VCL
21192: 162 unit uPSI_JvSimpleXml;                //JCL
21193: 163 unit uPSI_JclSimpleXml;               //JCL
21194: 164 unit uPSI_JvXmlDatabase;              //JCL
21195: 165 unit uPSI_JvMaxPixel;                 //JCL
21196: 166 unit uPSI_JvItemsSearchs;             //JCL
21197: 167 unit uPSI_StExpEng2;                  //SysTools4
21198: 168 unit uPSI_StGenLog;                   //SysTools4
21199: 169 unit uPSI_JvLogFile;                  //Jcl
21200: 170 unit uPSI_CPort;                      //ComPort Lib v4.11
21201: 171 unit uPSI_CPortCtl;                   //ComPort
21202: 172 unit uPSI_CPortEsc;                   //ComPort
21203: 173 unit BarCodeScaner;                   //ComPort
21204: 174 unit uPSI_JvGraph;                    //JCL
21205: 175 unit uPSI_JvComCtrls;                 //JCL
21206: 176 unit uPSI_GUITesting;                 //D Unit
21207: 177 unit uPSI_JvFindFiles;                //JCL
21208: 178 unit uPSI_StSystem;                   //SysTools4
21209: 179 unit uPSI_JvKeyboardStates;           //JCL
21210: 180 unit uPSI_JvMail;                     //JCL
21211: 181 unit uPSI_JclConsole;                 //JCL
21212: 182 unit uPSI_JclLANMan;                  //JCL
21213: 183 unit uPSI_IdCustomHTTPServer;         //Indy
21214: 184 unit IdHTTPServer                     //Indy
21215: 185 unit uPSI_IdTCPServer;                //Indy
21216: 186 unit uPSI_IdSocketHandle;             //Indy
21217: 187 unit uPSI_IdIOHandlerSocket;          //Indy
21218: 188 unit IdIOHandler;                     //Indy
21219: 189 unit uPSI_cutils;                     //Bloodshed
21220: 190 unit uPSI_BoldUtils;                  //boldsoft
21221: 191 unit uPSI_IdSimpleServer;             //Indy
21222: 192 unit uPSI_IdSSLOpenSSL;               //Indy
21223: 193 unit uPSI_IdMultipartFormData;        //Indy
21224: 194 unit uPSI_SynURIOpener;               //SynEdit
21225: 195 unit uPSI_PerlRegEx;                  //PCRE
21226: 196 unit uPSI_IdHeaderList;               //Indy
21227: 197 unit uPSI_StFirst;                    //SysTools4
21228: 198 unit uPSI_JvCtrls;                    //JCL
21229: 199 unit uPSI_IdTrivialFTPBase;           //Indy
21230: 200 unit uPSI_IdTrivialFTP;               //Indy
21231: 201 unit uPSI_IdUDPBase;                  //Indy
21232: 202 unit uPSI_IdUDPClient;                //Indy
21233: 203 unit uPSI_utypes;                     //for DMath.DLL
21234: 204 unit uPSI_ShellAPI;                   //Borland
21235: 205 unit uPSI_IdRemoteCMDClient;          //Indy
21236: 206 unit uPSI_IdRemoteCMDServer;          //Indy
21237: 207 unit IdRexecServer;                   //Indy
21238: 208 unit IdRexec; (unit uPSI_IdRexec;)    //Indy
```

```
21239: 209 unit IdUDPServer;                          //Indy
21240: 210 unit IdTimeUDPServer;                      //Indy
21241: 211 unit IdTimeServer;                         //Indy
21242: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)   //Indy
21243: 213 unit uPSI_IdIPWatch;                       //Indy
21244: 214 unit uPSI_IdIrcServer;                     //Indy
21245: 215 unit uPSI_IdMessageCollection;            //Indy
21246: 216 unit uPSI_cPEM;                            //Fundamentals 4
21247: 217 unit uPSI_cFundamentUtils;                 //Fundamentals 4
21248: 218 unit uPSI_uwinplot;                        //DMath
21249: 219 unit uPSI_xrtl_util_CPUUtils;              //ExtentedRTL
21250: 220 unit uPSI_GR32_System;                     //Graphics32
21251: 221 unit uPSI_cFileUtils;                      //Fundamentals 4
21252: 222 unit uPSI_cDateTime; (timemachine)         //Fundamentals 4
21253: 223 unit uPSI_cTimers; (high precision timer)  //Fundamentals 4
21254: 224 unit uPSI_cRandom;                         //Fundamentals 4
21255: 225 unit uPSI_ueval;                           //DMath
21256: 226 unit uPSI_xrtl_net_URIUtils;               //ExtendedRTL
21257: 227 unit xrtl_net_URIUtils;                    //ExtendedRTL
21258: 228 unit uPSI_ufft;  (FFT)                     //DMath
21259: 229 unit uPSI_DBXChannel;                      //Delphi
21260: 230 unit uPSI_DBXIndyChannel;                  //Delphi Indy
21261: 231 unit uPSI_xrtl_util_COMCat;                //ExtendedRTL
21262: 232 unit uPSI_xrtl_util_StrUtils;              //ExtendedRTL
21263: 233 unit uPSI_xrtl_util_VariantUtils;          //ExtendedRTL
21264: 234 unit uPSI_xrtl_util_FileUtils;             //ExtendedRTL
21265: 235 unit xrtl_util_Compat;                     //ExtendedRTL
21266: 236 unit uPSI_OleAuto;                         //Borland
21267: 237 unit uPSI_xrtl_util_COMUtils;              //ExtendedRTL
21268: 238 unit uPSI_CmAdmCtl;                        //Borland
21269: 239 unit uPSI_ValEdit2;                        //VCL
21270: 240 unit uPSI_GR32;  //Graphics32             //Graphics32
21271: 241 unit uPSI_GR32_Image;                      //Graphics32
21272: 242 unit uPSI_xrtl_util_TimeUtils;             //ExtendedRTL
21273: 243 unit uPSI_xrtl_util_TimeZone;              //ExtendedRTL
21274: 244 unit uPSI_xrtl_util_TimeStamp;             //ExtendedRTL
21275: 245 unit uPSI_xrtl_util_Map;                   //ExtendedRTL
21276: 246 unit uPSI_xrtl_util_Set;                   //ExtendedRTL
21277: 247 unit uPSI_CPortMonitor;                    //ComPort
21278: 248 unit uPSI_StIniStm;                        //SysTools4
21279: 249 unit uPSI_GR32_ExtImage;                   //Graphics32
21280: 250 unit uPSI_GR32_OrdinalMaps;                //Graphics32
21281: 251 unit uPSI_GR32_Rasterizers;                //Graphics32
21282: 252 unit uPSI_xrtl_util_Exception;             //ExtendedRTL
21283: 253 unit uPSI_xrtl_util_Value;                 //ExtendedRTL
21284: 254 unit uPSI_xrtl_util_Compare;               //ExtendedRTL
21285: 255 unit uPSI_FlatSB;                          //VCL
21286: 256 unit uPSI_JvAnalogClock;                   //JCL
21287: 257 unit uPSI_JvAlarms;                        //JCL
21288: 258 unit uPSI_JvSQLS;                          //JCL
21289: 259 unit uPSI_JvDBSecur;                       //JCL
21290: 260 unit uPSI_JvDBQBE;                         //JCL
21291: 261 unit uPSI_JvStarfield;                     //JCL
21292: 262 unit uPSI_JVCLMiscal;                      //JCL
21293: 263 unit uPSI_JvProfiler32;                    //JCL
21294: 264 unit uPSI_JvDirectories,                   //JCL
21295: 265 unit uPSI_JclSchedule,                     //JCL
21296: 266 unit uPSI_JclSvcCtrl,                      //JCL
21297: 267 unit uPSI_JvSoundControl,                  //JCL
21298: 268 unit uPSI_JvBDESQLScript,                  //JCL
21299: 269 unit uPSI_JvgDigits,                       //JCL>
21300: 270 unit uPSI_ImgList;                         //TCustomImageList
21301: 271 unit uPSI_JclMIDI;                         //JCL>
21302: 272 unit uPSI_JclWinMidi;                      //JCL>
21303: 273 unit uPSI_JclNTFS;                         //JCL>
21304: 274 unit uPSI_JclAppInst;                      //JCL>
21305: 275 unit uPSI_JvRle;                           //JCL>
21306: 276 unit uPSI_JvRas32;                         //JCL>
21307: 277 unit uPSI_JvImageDrawThread,               //JCL>
21308: 278 unit uPSI_JvImageWindow,                   //JCL>
21309: 279 unit uPSI_JvTransparentForm;               //JCL>
21310: 280 unit uPSI_JvWinDialogs;                    //JCL>
21311: 281 unit uPSI_JvSimLogic,                      //JCL>
21312: 282 unit uPSI_JvSimIndicator,                  //JCL>
21313: 283 unit uPSI_JvSimPID,                        //JCL>
21314: 284 unit uPSI_JvSimPIDLinker,                  //JCL>
21315: 285 unit uPSI_IdRFCReply;                      //Indy
21316: 286 unit uPSI_IdIdent;                         //Indy
21317: 287 unit uPSI_IdIdentServer;                   //Indy
21318: 288 unit uPSI_JvPatchFile;                     //JCL
21319: 289 unit uPSI_StNetPfm;                        //SysTools4
21320: 290 unit uPSI_StNet;                           //SysTools4
21321: 291 unit uPSI_JclPeImage;                      //JCL
21322: 292 unit uPSI_JclPrint;                        //JCL
21323: 293 unit uPSI_JclMime;                         //JCL
21324: 294 unit uPSI_JvRichEdit;                      //JCL
21325: 295 unit uPSI_JvDBRichEd;                      //JCL
21326: 296 unit uPSI_JvDice;                          //JCL
21327: 297 unit uPSI_JvFloatEdit;                     //JCL 3.9.8
```

```
21328: 298 unit uPSI_JvDirFrm;                      //JCL
21329: 299 unit uPSI_JvDualList;                    //JCL
21330: 300 unit uPSI_JvSwitch;                      ////JCL
21331: 301 unit uPSI_JvTimerLst;                    ////JCL
21332: 302 unit uPSI_JvMemTable;                    //JCL
21333: 303 unit uPSI_JvObjStr;                      //JCL
21334: 304 unit uPSI_StLArr;                        //SysTools4
21335: 305 unit uPSI_StWmDCpy;                      //SysTools4
21336: 306 unit uPSI_StText;                        //SysTools4
21337: 307 unit uPSI_StNTLog;                       //SysTools4
21338: 308 unit uPSI_xrtl_math_Integer;             //ExtendedRTL
21339: 309 unit uPSI_JvImagPrvw;                    //JCL
21340: 310 unit uPSI_JvFormPatch;                   //JCL
21341: 311 unit uPSI_JvPicClip;                     //JCL
21342: 312 unit uPSI_JvDataConv;                    //JCL
21343: 313 unit uPSI_JvCpuUsage;                    //JCL
21344: 314 unit uPSI_JclUnitConv_mX2;               //JCL
21345: 315 unit JvDualListForm;                     //JCL
21346: 316 unit uPSI_JvCpuUsage2;                   //JCL
21347: 317 unit uPSI_JvParserForm;                  //JCL
21348: 318 unit uPSI_JvJanTreeView;                 //JCL
21349: 319 unit uPSI_JvTransLED;                    //JCL
21350: 320 unit uPSI_JvPlaylist;                    //JCL
21351: 321 unit uPSI_JvFormAutoSize;                //JCL
21352: 322 unit uPSI_JvYearGridEditForm;            //JCL
21353: 323 unit uPSI_JvMarkupCommon;                //JCL
21354: 324 unit uPSI_JvChart;                       //JCL
21355: 325 unit uPSI_JvXPCore;                      //JCL
21356: 326 unit uPSI_JvXPCoreUtils;                 //JCL
21357: 327 unit uPSI_StatsClasses;                  //mX4
21358: 328 unit uPSI_ExtCtrls2;                     //VCL
21359: 329 unit uPSI_JvUrlGrabbers;                 //JCL
21360: 330 unit uPSI_JvXmlTree;                     //JCL
21361: 331 unit uPSI_JvWavePlayer;                  //JCL
21362: 332 unit uPSI_JvUnicodeCanvas;               //JCL
21363: 333 unit uPSI_JvTFUtils;                     //JCL
21364: 334 unit uPSI_IdServerIOHandler;             //Indy
21365: 335 unit uPSI_IdServerIOHandlerSocket;       //Indy
21366: 336 unit uPSI_IdMessageCoder;                //Indy
21367: 337 unit uPSI_IdMessageCoderMIME;            //Indy
21368: 338 unit uPSI_IdMIMETypes;                   //Indy
21369: 339 unit uPSI_JvConverter;                   //JCL
21370: 340 unit uPSI_JvCsvParse;                    //JCL
21371: 341 unit uPSI_umath;  unit uPSI_ugamma;      //DMath
21372: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
21373: 343 unit uPSI_JvDBGridExport;                //JCL
21374: 344 unit uPSI_JvgExport;                     //JCL
21375: 345 unit uPSI_JvSerialMaker;                 //JCL
21376: 346 unit uPSI_JvWin32;                       //JCL
21377: 347 unit uPSI_JvPaintFX;                     //JCL
21378: 348 unit uPSI_JvOracleDataSet; (beta)        //JCL
21379: 349 unit uPSI_JvValidators; (preview)        //JCL
21380: 350 unit uPSI_JvNTEventLog;                  //JCL
21381: 351 unit uPSI_ShellZipTool;                  //mX4
21382: 352 unit uPSI_JvJoystick;                    //JCL
21383: 353 unit uPSI_JvMailSlots;                   //JCL
21384: 354 unit uPSI_JclComplex;                    //JCL
21385: 355 unit uPSI_SynPdf;                        //Synopse
21386: 356 unit uPSI_Registry;                      //VCL
21387: 357 unit uPSI_TlHelp32;                      //VCL
21388: 358 unit uPSI_JclRegistry;                   //JCL
21389: 359 unit uPSI_JvAirBrush;                    //JCL
21390: 360 unit uPSI_mORMotReport;                  //Synopse
21391: 361 unit uPSI_JclLocales;                    //JCL
21392: 362 unit uPSI_SynEdit;                       //SynEdit
21393: 363 unit uPSI_SynEditTypes;                  //SynEdit
21394: 364 unit uPSI_SynMacroRecorder;              //SynEdit
21395: 365 unit uPSI_LongIntList;                   //SynEdit
21396: 366 unit uPSI_devcutils;                     //DevC
21397: 367 unit uPSI_SynEditMiscClasses;            //SynEdit
21398: 368 unit uPSI_SynEditRegexSearch;            //SynEdit
21399: 369 unit uPSI_SynEditHighlighter;            //SynEdit
21400: 370 unit uPSI_SynHighlighterPas;             //SynEdit
21401: 371 unit uPSI_JvSearchFiles;                 //JCL
21402: 372 unit uPSI_SynHighlighterAny;             //Lazarus
21403: 373 unit uPSI_SynEditKeyCmds;                //SynEdit
21404: 374 unit uPSI_SynEditMiscProcs,              //SynEdit
21405: 375 unit uPSI_SynEditKbdHandler              //SynEdit
21406: 376 unit uPSI_JvAppInst,                     //JCL
21407: 377 unit uPSI_JvAppEvent;                    //JCL
21408: 378 unit uPSI_JvAppCommand;                  //JCL
21409: 379 unit uPSI_JvAnimTitle;                   //JCL
21410: 380 unit uPSI_JvAnimatedImage;               //JCL
21411: 381 unit uPSI_SynEditExport;                 //SynEdit
21412: 382 unit uPSI_SynExportHTML;                 //SynEdit
21413: 383 unit uPSI_SynExportRTF;                  //SynEdit
21414: 384 unit uPSI_SynEditSearch;                 //SynEdit
21415: 385 unit uPSI_fMain_back                     //maXbox;
21416: 386 unit uPSI_JvZoom;                        //JCL
```

```
21417: 387 unit uPSI_PMrand;                        //PM
21418: 388 unit uPSI_JvSticker;                     //JCL
21419: 389 unit uPSI_XmlVerySimple;                 //mX4
21420: 390 unit uPSI_Services;                      //ExtPascal
21421: 391 unit uPSI_ExtPascalUtils;                //ExtPascal
21422: 392 unit uPSI_SocketsDelphi;                 //ExtPascal
21423: 393 unit uPSI_StBarC;                        //SysTools
21424: 394 unit uPSI_StDbBarC;                      //SysTools
21425: 395 unit uPSI_StBarPN;                       //SysTools
21426: 396 unit uPSI_StDbPNBC;                      //SysTools
21427: 397 unit uPSI_StDb2DBC;                      //SysTools
21428: 398 unit uPSI_StMoney;                       //SysTools
21429: 399 unit uPSI_JvForth;                       //JCL
21430: 400 unit uPSI_RestRequest;                   //mX4
21431: 401 unit uPSI_HttpRESTConnectionIndy;        //mX4
21432: 402 unit uPSI_JvXmlDatabase; //update       //JCL
21433: 403 unit uPSI_StAstro;                       //SysTools
21434: 404 unit uPSI_StSort;                        //SysTools
21435: 405 unit uPSI_StDate;                        //SysTools
21436: 406 unit uPSI_StDateSt;                      //SysTools
21437: 407 unit uPSI_StBase;                        //SysTools
21438: 408 unit uPSI_StVInfo;                       //SysTools
21439: 409 unit uPSI_JvBrowseFolder;                //JCL
21440: 410 unit uPSI_JvBoxProcs;                    //JCL
21441: 411 unit uPSI_urandom; (unit uranuvag;)      //DMath
21442: 412 unit uPSI_usimann; (unit ugenalg;)       //DMath
21443: 413 unit uPSI_JvHighlighter;                 //JCL
21444: 414 unit uPSI_Diff;                          //mX4
21445: 415 unit uPSI_SpringWinAPI;                  //DSpring
21446: 416 unit uPSI_StBits;                        //SysTools
21447: 417 unit uPSI_TomDBQue;                      //mX4
21448: 418 unit uPSI_MultilangTranslator;           //mX4
21449: 419 unit uPSI_HyperLabel;                    //mX4
21450: 420 unit uPSI_Starter;                       //mX4
21451: 421 unit uPSI_FileAssocs;                    //devC
21452: 422 unit uPSI_devFileMonitorX;               //devC
21453: 423 unit uPSI_devrun;                        //devC
21454: 424 unit uPSI_devExec;                       //devC
21455: 425 unit uPSI_oysUtils;                      //devC
21456: 426 unit uPSI_DosCommand;                    //devC
21457: 427 unit uPSI_CppTokenizer;                  //devC
21458: 428 unit uPSI_JvHLParser;                    //devC
21459: 429 unit uPSI_JclMapi;                       //JCL
21460: 430 unit uPSI_JclShell;                      //JCL
21461: 431 unit uPSI_JclCOM;                        //JCL
21462: 432 unit uPSI_GR32_Math;                     //Graphics32
21463: 433 unit uPSI_GR32_LowLevel;                 //Graphics32
21464: 434 unit uPSI_SimpleHl;                      //mX4
21465: 435 unit uPSI_GR32_Filters,                  //Graphics32
21466: 436 unit uPSI_GR32_VectorMaps;               //Graphics32
21467: 437 unit uPSI_cXMLFunctions;                 //Fundamentals 4
21468: 438 unit uPSI_JvTimer;                       //JCL
21469: 439 unit uPSI_cHTTPUtils;                    //Fundamentals 4
21470: 440 unit uPSI_cTLSUtils;                     //Fundamentals 4
21471: 441 unit uPSI_JclGraphics;                   //JCL
21472: 442 unit uPSI_JclSynch;                      //JCL
21473: 443 unit uPSI_IdTelnet;                      //Indy
21474: 444 unit uPSI_IdTelnetServer,                //Indy
21475: 445 unit uPSI_IdEcho,                        //Indy
21476: 446 unit uPSI_IdEchoServer,                  //Indy
21477: 447 unit uPSI_IdEchoUDP,                     //Indy
21478: 448 unit uPSI_IdEchoUDPServer,               //Indy
21479: 449 unit uPSI_IdSocks,                       //Indy
21480: 450 unit uPSI_IdAntiFreezeBase;              //Indy
21481: 451 unit uPSI_IdHostnameServer;              //Indy
21482: 452 unit uPSI_IdTunnelCommon,                //Indy
21483: 453 unit uPSI_IdTunnelMaster,                //Indy
21484: 454 unit uPSI_IdTunnelSlave,                 //Indy
21485: 455 unit uPSI_IdRSH,                         //Indy
21486: 456 unit uPSI_IdRSHServer,                   //Indy
21487: 457 unit uPSI_Spring_Cryptography_Utils;     //Spring4Delphi
21488: 458 unit uPSI_MapReader,                     //devC
21489: 459 unit uPSI_LibTar,                        //devC
21490: 460 unit uPSI_IdStack;                       //Indy
21491: 461 unit uPSI_IdBlockCipherIntercept;        //Indy
21492: 462 unit uPSI_IdChargenServer;               //Indy
21493: 463 unit uPSI_IdFTPServer,                   //Indy
21494: 464 unit uPSI_IdException,                   //Indy
21495: 465 unit uPSI_utexplot;                      //DMath
21496: 466 unit uPSI_uwinstr;                       //DMath
21497: 467 unit uPSI_VarRecUtils;                   //devC
21498: 468 unit uPSI_JvStringListToHtml,            //JCL
21499: 469 unit uPSI_JvStringHolder,                //JCL
21500: 470 unit uPSI_IdCoder;                       //Indy
21501: 471 unit uPSI_SynHighlighterDfm;             //Synedit
21502: 472 unit uHighlighterProcs; in 471           //Synedit
21503: 473 unit uPSI_LazFileUtils,                  //LCL
21504: 474 unit uPSI_IDECmdLine;                    //LCL
21505: 475 unit uPSI_lazMasks;                      //LCL
```

```
21506: 476 unit uPSI_ip_misc;                      //mX4
21507: 477 unit uPSI_Barcode;                      //LCL
21508: 478 unit uPSI_SimpleXML;                    //LCL
21509: 479 unit uPSI_JclIniFiles;                  //JCL
21510: 480 unit uPSI_D2XXUnit;  {$X-}              //FTDI
21511: 481 unit uPSI_JclDateTime;                  //JCL
21512: 482 unit uPSI_JclEDI;                       //JCL
21513: 483 unit uPSI_JclMiscel2;                   //JCL
21514: 484 unit uPSI_JclValidation;                //JCL
21515: 485 unit uPSI_JclAnsiStrings; {-PString}    //JCL
21516: 486 unit uPSI_SynEditMiscProcs2;            //Synedit
21517: 487 unit uPSI_JclStreams;                   //JCL
21518: 488 unit uPSI_QRCode;                       //mX4
21519: 489 unit uPSI_BlockSocket;                  //ExtPascal
21520: 490 unit uPSI_Masks,Utils                   //VCL
21521: 491 unit uPSI_synautil;                     //Synapse!
21522: 492 unit uPSI_JclMath_Class;                //JCL RTL
21523: 493 unit ugamdist; //Gamma function        //DMath
21524: 494 unit uibeta, ucorrel; //IBeta          //DMath
21525: 495 unit uPSI_SRMgr;                        //mX4
21526: 496 unit uPSI_HotLog;                       //mX4
21527: 497 unit uPSI_DebugBox;                     //mX4
21528: 498 unit uPSI_ustrings;                     //DMath
21529: 499 unit uPSI_uregtest;                     //DMath
21530: 500 unit uPSI_usimplex;                     //DMath
21531: 501 unit uPSI_uhyper;                       //DMath
21532: 502 unit uPSI_IdHL7;                        //Indy
21533: 503 unit uPSI_IdIPMCastBase,                //Indy
21534: 504 unit uPSI_IdIPMCastServer;              //Indy
21535: 505 unit uPSI_IdIPMCastClient;              //Indy
21536: 506 unit uPSI_unlfit; //nlregression       //DMath
21537: 507 unit uPSI_IdRawHeaders;                 //Indy
21538: 508 unit uPSI_IdRawClient;                  //Indy
21539: 509 unit uPSI_IdRawFunctions;               //Indy
21540: 510 unit uPSI_IdTCPStream;                  //Indy
21541: 511 unit uPSI_IdSNPP;                       //Indy
21542: 512 unit uPSI_St2DBarC;                     //SysTools
21543: 513 unit uPSI_ImageWin;  //FTL             //VCL
21544: 514 unit uPSI_CustomDrawTreeView; //FTL     //VCL
21545: 515 unit uPSI_GraphWin;  //FTL             //VCL
21546: 516 unit uPSI_actionMain;  //FTL           //VCL
21547: 517 unit uPSI_StSpawn;                      //SysTools
21548: 518 unit uPSI_CtlPanel;                     //VCL
21549: 519 unit uPSI_IdLPR;                        //Indy
21550: 520 unit uPSI_SockRequestInterpreter;       //Indy
21551: 521 unit uPSI_ulambert;                     //DMath
21552: 522 unit uPSI_ucholesk;                     //DMath
21553: 523 unit uPSI_SimpleDS;                     //VCL
21554: 524 unit uPSI_DBXSqlScanner;                //VCL
21555: 525 unit uPSI_DBXMetaDataUtil;              //VCL
21556: 526 unit uPSI_Chart;                        //TEE
21557: 527 unit uPSI_TeeProcs;                     //TEE
21558: 528 unit mXBDEUtils;                        //mX4
21559: 529 unit uPSI_MDIEdit;                      //VCL
21560: 530 unit uPSI_CopyPrsr;                     //VCL
21561: 531 unit uPSI_SockApp;                      //VCL
21562: 532 unit uPSI_AppEvnts;                     //VCL
21563: 533 unit uPSI_ExtActns;                     //VCL
21564: 534 unit uPSI_TeEngine;                     //TEE
21565: 535 unit uPSI_CoolMain; //browser          //VCL
21566: 536 unit uPSI_StCRC;                        //SysTools
21567: 537 unit uPSI_StDecMth2;                    //SysTools
21568: 538 unit uPSI_frmExportMain;                //Synedit
21569: 539 unit uPSI_SynDBEdit;                    //Synedit
21570: 540 unit uPSI_SynEditWildcardSearch;        //Synedit
21571: 541 unit uPSI_BoldComUtils;                 //BOLD
21572: 542 unit uPSI_BoldIsoDateTime;              //BOLD
21573: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils  //BOLD
21574: 544 unit uPSI_BoldXMLRequests;              //BOLD
21575: 545 unit uPSI_BoldStringList;               //BOLD
21576: 546 unit uPSI_BoldFileHandler;              //BOLD
21577: 547 unit uPSI_BoldContainers;               //BOLD
21578: 548 unit uPSI_BoldQueryUserDlg;             //BOLD
21579: 549 unit uPSI_BoldWinINet;                  //BOLD
21580: 550 unit uPSI_BoldQueue;                    //BOLD
21581: 551 unit uPSI_JvPcx;                        //JCL
21582: 552 unit uPSI_IdWhois;                      //Indy
21583: 553 unit uPSI_IdWhoIsServer;                //Indy
21584: 554 unit uPSI_IdGopher;                     //Indy
21585: 555 unit uPSI_IdDateTimeStamp;              //Indy
21586: 556 unit uPSI_IdDayTimeServer;              //Indy
21587: 557 unit uPSI_IdDayTimeUDP;                 //Indy
21588: 558 unit uPSI_IdDayTimeUDPServer;           //Indy
21589: 559 unit uPSI_IdDICTServer;                 //Indy
21590: 560 unit uPSI_IdDiscardServer;              //Indy
21591: 561 unit uPSI_IdDiscardUDPServer;           //Indy
21592: 562 unit uPSI_IdMappedFTP;                  //Indy
21593: 563 unit uPSI_IdMappedPortTCP;              //Indy
21594: 564 unit uPSI_IdGopherServer;               //Indy
```

```
21595: 565 unit uPSI_IdQotdServer;                  //Indy
21596: 566 unit uPSI_JvRgbToHtml;                   //JCL
21597: 567 unit uPSI_JvRemLog,                      //JCL
21598: 568 unit uPSI_JvSysComp;                    //JCL
21599: 569 unit uPSI_JvTMTL;                       //JCL
21600: 570 unit uPSI_JvWinampAPI;                  //JCL
21601: 571 unit uPSI_MSysUtils;                    //mX4
21602: 572 unit uPSI_ESBMaths;                     //ESB
21603: 573 unit uPSI_ESBMaths2;                    //ESB
21604: 574 unit uPSI_uLkJSON;                      //Lk
21605: 575 unit uPSI_ZURL;  //Zeos                 //Zeos
21606: 576 unit uPSI_ZSysUtils;                    //Zeos
21607: 577 unit unaUtils internals                 //UNA
21608: 578 unit uPSI_ZMatchPattern;                //Zeos
21609: 579 unit uPSI_ZClasses;                     //Zeos
21610: 580 unit uPSI_ZCollections;                 //Zeos
21611: 581 unit uPSI_ZEncoding;                    //Zeos
21612: 582 unit uPSI_IdRawBase;                    //Indy
21613: 583 unit uPSI_IdNTLM;                       //Indy
21614: 584 unit uPSI_IdNNTP;                       //Indy
21615: 585 unit uPSI_usniffer; //PortScanForm      //mX4
21616: 586 unit uPSI_IdCoderMIME;                  //Indy
21617: 587 unit uPSI_IdCoderUUE;                   //Indy
21618: 588 unit uPSI_IdCoderXXE;                   //Indy
21619: 589 unit uPSI_IdCoder3to4;                  //Indy
21620: 590 unit uPSI_IdCookie;                     //Indy
21621: 591 unit uPSI_IdCookieManager;              //Indy
21622: 592 unit uPSI_WDosSocketUtils;              //WDos
21623: 593 unit uPSI_WDosPlcUtils;                 //WDos
21624: 594 unit uPSI_WDosPorts;                    //WDos
21625: 595 unit uPSI_WDosResolvers;                //WDos
21626: 596 unit uPSI_WDosTimers;                   //WDos
21627: 597 unit uPSI_WDosPlcs;                     //WDos
21628: 598 unit uPSI_WDosPneumatics;               //WDos
21629: 599 unit uPSI_IdFingerServer;               //Indy
21630: 600 unit uPSI_IdDNSResolver;                //Indy
21631: 601 unit uPSI_IdHTTPWebBrokerBridge;        //Indy
21632: 602 unit uPSI_IdIntercept;                  //Indy
21633: 603 unit uPSI_IdIPMCastBase;                //Indy
21634: 604 unit uPSI_IdLogBase;                    //Indy
21635: 605 unit uPSI_IdIOHandlerStream;            //Indy
21636: 606 unit uPSI_IdMappedPortUDP;              //Indy
21637: 607 unit uPSI_IdQOTDUDPServer;              //Indy
21638: 608 unit uPSI_IdQOTDUDP;                    //Indy
21639: 609 unit uPSI_IdSysLog;                     //Indy
21640: 610 unit uPSI_IdSysLogServer;               //Indy
21641: 611 unit uPSI_IdSysLogMessage;              //Indy
21642: 612 unit uPSI_IdTimeServer;                 //Indy
21643: 613 unit uPSI_IdTimeUDP;                    //Indy
21644: 614 unit uPSI_IdTimeUDPServer;              //Indy
21645: 615 unit uPSI_IdUserAccounts;               //Indy
21646: 616 unit uPSI_TextUtils;                    //mX4
21647: 617 unit uPSI_MandelbrotEngine;             //mX4
21648: 618 unit uPSI_delphi_arduino_Unit1;         //mX4
21649: 619 unit uPSI_DTDSchema2;                   //mX4
21650: 620 unit uPSI_fplotMain;                    //DMath
21651: 621 unit uPSI_FindFileIter;                 //mX4
21652: 622 unit uPSI_PppState;  (JclStrHashMap)    //PPP
21653: 623 unit uPSI_PppParser;                    //PPP
21654: 624 unit uPSI_PppLexer;                     //PPP
21655: 625 unit uPSI_PCharUtils;                   //PPP
21656: 626 unit uPSI_uJSON;                        //WU
21657: 627 unit uPSI_JclStrHashMap;                //JCL
21658: 628 unit uPSI_JclHookExcept;                //JCL
21659: 629 unit uPSI_EncdDecd;                     //VCL
21660: 630 unit uPSI_SockAppReg;                   //VCL
21661: 631 unit uPSI_PJFileHandle;                 //PJ
21662: 632 unit uPSI_PJEnvVars;                    //PJ
21663: 633 unit uPSI_PJPipe;                       //PJ
21664: 634 unit uPSI_PJPipeFilters;                //PJ
21665: 635 unit uPSI_PJConsoleApp;                 //PJ
21666: 636 unit uPSI_UConsoleAppEx;                //PJ
21667: 637 unit uPSI_DbxSocketChannelNative,       //VCL
21668: 638 unit uPSI_DbxDataGenerator,             //VCL
21669: 639 unit uPSI_DBXClient;                    //VCL
21670: 640 unit uPSI_IdLogEvent;                   //Indy
21671: 641 unit uPSI_Reversi;                      //mX4
21672: 642 unit uPSI_Geometry;                     //mX4
21673: 643 unit uPSI_IdSMTPServer;                 //Indy
21674: 644 unit uPSI_Textures;                     //mX4
21675: 645 unit uPSI_IBX;                          //VCL
21676: 646 unit uPSI_IWDBCommon;                   //VCL
21677: 647 unit uPSI_SortGrid;                     //mX4
21678: 648 unit uPSI_IB;                           //VCL
21679: 649 unit uPSI_IBScript;                     //VCL
21680: 650 unit uPSI_JvCSVBaseControls;            //JCL
21681: 651 unit uPSI_Jvg3DColors;                  //JCL
21682: 652 unit uPSI_JvHLEditor;  //beat           //JCL
21683: 653 unit uPSI_JvShellHook;                  //JCL
```

```
21684: 654 unit uPSI_DBCommon2                    //VCL
21685: 655 unit uPSI_JvSHFileOperation;           //JCL
21686: 656 unit uPSI_uFilexport;                  //mX4
21687: 657 unit uPSI_JvDialogs;                   //JCL
21688: 658 unit uPSI_JvDBTreeView;                //JCL
21689: 659 unit uPSI_JvDBUltimGrid;               //JCL
21690: 660 unit uPSI_JvDBQueryParamsForm;         //JCL
21691: 661 unit uPSI_JvExControls;                //JCL
21692: 662 unit uPSI_JvBDEMemTable;               //JCL
21693: 663 unit uPSI_JvCommStatus;                //JCL
21694: 664 unit uPSI_JvMailSlots2;                //JCL
21695: 665 unit uPSI_JvgWinMask;                  //JCL
21696: 666 unit uPSI_StEclpse;                    //SysTools
21697: 667 unit uPSI_StMime;                      //SysTools
21698: 668 unit uPSI_StList;                      //SysTools
21699: 669 unit uPSI_StMerge;                     //SysTools
21700: 670 unit uPSI_StStrS;                      //SysTools
21701: 671 unit uPSI_StTree,                      //SysTools
21702: 672 unit uPSI_StVArr;                      //SysTools
21703: 673 unit uPSI_StRegIni;                    //SysTools
21704: 674 unit uPSI_urkf;                        //DMath
21705: 675 unit uPSI_usvd;                        //DMath
21706: 676 unit uPSI_DepWalkUtils;                //JCL
21707: 677 unit uPSI_OptionsFrm;                  //JCL
21708: 678 unit yuvconverts;                      //mX4
21709: 679 uPSI_JvPropAutoSave;                   //JCL
21710: 680 uPSI_AclAPI;                           //alcinoe
21711: 681 uPSI_AviCap;                           //alcinoe
21712: 682 uPSI_ALAVLBinaryTree;                  //alcinoe
21713: 683 uPSI_ALFcnMisc;                        //alcinoe
21714: 684 uPSI_ALStringList;                     //alcinoe
21715: 685 uPSI_ALQuickSortList;                  //alcinoe
21716: 686 uPSI_ALStaticText;                     //alcinoe
21717: 687 uPSI_ALJSONDoc;                        //alcinoe
21718: 688 uPSI_ALGSMComm;                        //alcinoe
21719: 689 uPSI_ALWindows;                        //alcinoe
21720: 690 uPSI_ALMultiPartFormDataParser;        //alcinoe
21721: 691 uPSI_ALHttpCommon;                     //alcinoe
21722: 692 uPSI_ALWebSpider,                      //alcinoe
21723: 693 uPSI_ALHttpClient;                     //alcinoe
21724: 694 uPSI_ALFcnHTML;                        //alcinoe
21725: 695 uPSI_ALFTPClient;                      //alcinoe
21726: 696 uPSI_ALInternetMessageCommon;          //alcinoe
21727: 697 uPSI_ALWininetHttpClient;              //alcinoe
21728: 698 uPSI_ALWinInetFTPClient;               //alcinoe
21729: 699 uPSI_ALWinHttpWrapper;                 //alcinoe
21730: 700 uPSI_ALWinHttpClient;                  //alcinoe
21731: 701 uPSI_ALFcnWinSock;                     //alcinoe
21732: 702 uPSI_ALFcnSQL;                         //alcinoe
21733: 703 uPSI_ALFcnCGI;                         //alcinoe
21734: 704 uPSI_ALFcnExecute;                     //alcinoe
21735: 705 uPSI_ALFcnFile;                        //alcinoe
21736: 706 uPSI_ALFcnMime;                        //alcinoe
21737: 707 uPSI_ALPhpRunner;                      //alcinoe
21738: 708 uPSI_ALGraphic;                        //alcinoe
21739: 709 uPSI_ALIniFiles;                       //alcinoe
21740: 710 uPSI_ALMemCachedClient;                //alcinoe
21741: 711 unit uPSI_MyGrids;                     //mX4
21742: 712 uPSI_ALMultiPartMixedParser            //alcinoe
21743: 713 uPSI_ALSMTPClient                      //alcinoe
21744: 714 uPSI_ALNNTPClient;                     //alcinoe
21745: 715 uPSI_ALHintBalloon;                    //alcinoe
21746: 716 unit uPSI_ALXmlDoc;                    //alcinoe
21747: 717 unit uPSI_IPCThrd;                     //VCL
21748: 718 unit uPSI_MonForm;                     //VCL
21749: 719 unit uPSI_TeCanvas;                    //Orpheus
21750: 720 unit uPSI_Ovcmisc;                     //Orpheus
21751: 721 unit uPSI_ovcfiler;                    //Orpheus
21752: 722 unit uPSI_ovcstate;                    //Orpheus
21753: 723 unit uPSI_ovccoco;                     //Orpheus
21754: 724 unit uPSI_ovcrvexp;                    //Orpheus
21755: 725 unit uPSI_OvcFormatSettings;           //Orpheus
21756: 726 unit uPSI_OvcUtils;                    //Orpheus
21757: 727 unit uPSI_ovcstore;                    //Orpheus
21758: 728 unit uPSI_ovcstr;                      //Orpheus
21759: 729 unit uPSI_ovcmru;                      //Orpheus
21760: 730 unit uPSI_ovccmd;                      //Orpheus
21761: 731 unit uPSI_ovctimer;                    //Orpheus
21762: 732 unit uPSI_ovcintl;                     //Orpheus
21763: 733 uPSI_AfCircularBuffer;                 //AsyncFree
21764: 734 uPSI_AfUtils;                          //AsyncFree
21765: 735 uPSI_AfSafeSync;                       //AsyncFree
21766: 736 uPSI_AfComPortCore;                    //AsyncFree
21767: 737 uPSI_AfComPort;                        //AsyncFree
21768: 738 uPSI_AfPortControls;                   //AsyncFree
21769: 739 uPSI_AfDataDispatcher;                 //AsyncFree
21770: 740 uPSI_AfViewers;                        //AsyncFree
21771: 741 uPSI_AfDataTerminal;                   //AsyncFree
21772: 742 uPSI_SimplePortMain;                   //AsyncFree
```

```
21773: 743 unit uPSI_ovcclock;                    //Orpheus
21774: 744 unit uPSI_o32intlst;                   //Orpheus
21775: 745 unit uPSI_o32ledlabel;                 //Orpheus
21776: 746 unit uPSI_AlMySqlClient;               //alcinoe
21777: 747 unit uPSI_ALFBXClient;                 //alcinoe
21778: 748 unit uPSI_ALFcnSQL;                    //alcinoe
21779: 749 unit uPSI_AsyncTimer;                  //mX4
21780: 750 unit uPSI_ApplicationFileIO;           //mX4
21781: 751 unit uPSI_PsAPI;                       //VCLé
21782: 752 uPSI_ovcuser;                          //Orpheus
21783: 753 uPSI_ovcurl;                           //Orpheus
21784: 754 uPSI_ovcvlb;                           //Orpheus
21785: 755 uPSI_ovccolor;                         //Orpheus
21786: 756 uPSI_ALFBXLib,                         //alcinoe
21787: 757 uPSI_ovcmeter;                         //Orpheus
21788: 758 uPSI_ovcpeakm;                         //Orpheus
21789: 759 uPSI_O32BGSty;                         //Orpheus
21790: 760 uPSI_ovcBidi;                          //Orpheus
21791: 761 uPSI_ovctcary;                         //Orpheus
21792: 762 uPSI_DXPUtils;                         //mX4
21793: 763 uPSI_ALMultiPartBaseParser;            //alcinoe
21794: 764 uPSI_ALMultiPartAlternativeParser;     //alcinoe
21795: 765 uPSI_ALPOP3Client;                     //alcinoe
21796: 766 uPSI_SmallUtils;                       //mX4
21797: 767 uPSI_MakeApp;                          //mX4
21798: 768 uPSI_O32MouseMon;                      //Orpheus
21799: 769 uPSI_OvcCache;                         //Orpheus
21800: 770 uPSI_ovccalc;                          //Orpheus
21801: 771 uPSI_Joystick,                         //OpenGL
21802: 772 uPSI_ScreenSaver;                      //OpenGL
21803: 773 uPSI_XCollection,                      //OpenGL
21804: 774 uPSI_Polynomials,                      //OpenGL
21805: 775 uPSI_PersistentClasses, //9.86        //OpenGL
21806: 776 uPSI_VectorLists;                      //OpenGL
21807: 777 uPSI_XOpenGL,                         //OpenGL
21808: 778 uPSI_MeshUtils;                        //OpenGL
21809: 779 unit uPSI_JclSysUtils;                 //JCL
21810: 780 unit uPSI_JclBorlandTools;             //JCL
21811: 781 unit JclFileUtils_max;                 //JCL
21812: 782 uPSI_AfDataControls,                   //AsyncFree
21813: 783 uPSI_GLSilhouette;                     //OpenGL
21814: 784 uPSI_JclSysUtils_class;                //JCL
21815: 785 uPSI_JclFileUtils_class;               //JCL
21816: 786 uPSI_FileUtil;                         //JCL
21817: 787 uPSI_changefind;                       //mX4
21818: 788 uPSI_cmdIntf;                          //mX4
21819: 789 uPSI_fservice;                         //mX4
21820: 790 uPSI_Keyboard;                         //OpenGL
21821: 791 uPSI_VRMLParser,                       //OpenGL
21822: 792 uPSI_GLFileVRML,                       //OpenGL
21823: 793 uPSI_Octree;                           //OpenGL
21824: 794 uPSI_GLPolyhedron,                     //OpenGL
21825: 795 uPSI_GLCrossPlatform;                  //OpenGL
21826: 796 uPSI_GLParticles;                      //OpenGL
21827: 797 uPSI_GLNavigator;                      //OpenGL
21828: 798 uPSI_GLStarRecord;                     //OpenGL
21829: 799 uPSI_GLTextureCombiners;               //OpenGL
21830: 800 uPSI_GLCanvas;                         //OpenGL
21831: 801 uPSI_GeometryBB;                       //OpenGL
21832: 802 uPSI_GeometryCoordinates;              //OpenGL
21833: 803 uPSI_VectorGeometry;                   //OpenGL
21834: 804 uPSI_BumpMapping;                      //OpenGL
21835: 805 uPSI_TGA;                              //OpenGL
21836: 806 uPSI_GLVectorFileObjects;              //OpenGL
21837: 807 uPSI_IMM;                              //VCL
21838: 808 uPSI_CategoryButtons;                  //VCL
21839: 809 uPSI_ButtonGroup;                      //VCL
21840: 810 uPSI_DbExcept;                         //VCL
21841: 811 uPSI_AxCtrls;                          //VCL
21842: 812 uPSI_GL_actorUnit1;                    //OpenGL
21843: 813 uPSI_StdVCL;                           //VCL
21844: 814 unit CurvesAndSurfaces;                //OpenGL
21845: 815 uPSI_DataAwareMain;                    //AsyncFree
21846: 816 uPSI_TabNotBk;                         //VCL
21847: 817 uPSI_udwsfiler;                        //mX4
21848: 818 uPSI_synaip;                           //Synapse!
21849: 819 uPSI_synacode;                         //Synapse
21850: 820 uPSI_synachar;                         //Synapse
21851: 821 uPSI_synamisc;                         //Synapse
21852: 822 uPSI_synaser;                          //Synapse
21853: 823 uPSI_synaicnv;                         //Synapse
21854: 824 uPSI_tlntsend;                         //Synapse
21855: 825 uPSI_pingsend;                         //Synapse
21856: 826 uPSI_blcksock;                         //Synapse
21857: 827 uPSI_asn1util;                         //Synapse
21858: 828 uPSI_dnssend;                          //Synapse
21859: 829 uPSI_clamsend;                         //Synapse
21860: 830 uPSI_ldapsend;                         //Synapse
21861: 831 uPSI_mimemess;                         //Synapse
```

```
21862: 832 uPSI_slogsend;                          //Synapse
21863: 833 uPSI_mimepart;                          //Synapse
21864: 834 uPSI_mimeinln;                          //Synapse
21865: 835 uPSI_ftpsend,                          //Synapse
21866: 836 uPSI_ftptsend;                         //Synapse
21867: 837 uPSI_httpsend;                         //Synapse
21868: 838 uPSI_sntpsend;                         //Synapse
21869: 839 uPSI_smtpsend;                         //Synapse
21870: 840 uPSI_snmpsend;                         //Synapse
21871: 841 uPSI_imapsend;                         //Synapse
21872: 842 uPSI_pop3send;                         //Synapse
21873: 843 uPSI_nntpsend;                         //Synapse
21874: 844 uPSI_ssl_cryptlib;                     //Synapse
21875: 845 uPSI_ssl_openssl;                      //Synapse
21876: 846 uPSI_synhttp_daemon;                   //Synapse
21877: 847 uPSI_NetWork;                         //mX4
21878: 848 uPSI_PingThread;                      //Synapse
21879: 849 uPSI_JvThreadTimer;                   //JCL
21880: 850 unit uPSI_wwSystem;                   //InfoPower
21881: 851 unit uPSI_IdComponent;                //Indy
21882: 852 unit uPSI_IdIOHandlerThrottle;        //Indy
21883: 853 unit uPSI_Themes;                     //VCL
21884: 854 unit uPSI_StdStyleActnCtrls;          //VCL
21885: 855 unit uPSI_UDDIHelper;                 //VCL
21886: 856 unit uPSI_IdIMAP4Server;              //Indy
21887: 857 uPSI_VariantSymbolTable,              //VCL //3.9.9.92
21888: 858 uPSI_udf_glob,                        //mX4
21889: 859 uPSI_TabGrid,                         //VCL
21890: 860 uPSI_JsDBTreeView,                    //mX4
21891: 861 uPSI_JsSendMail,                      //mX4
21892: 862 uPSI_dbTvRecordList,                  //mX4
21893: 863 uPSI_TreeVwEx,                        //mX4
21894: 864 uPSI_ECDataLink,                      //mX4
21895: 865 uPSI_dbTree,                          //mX4
21896: 866 uPSI_dbTreeCBox,                      //mX4
21897: 867 unit uPSI_Debug; //TfrmDebug         //mX4
21898: 868 uPSI_TimeFncs;                        //mX4
21899: 869 uPSI_FileIntf,                       //VCL
21900: 870 uPSI_SockTransport,                  //RTL
21901: 871 unit uPSI_WinInet;                   //RTL
21902: 872 unit uPSI_Wwstr;                     //mX4
21903: 873 uPSI_DBLookup,                       //VCL
21904: 874 uPSI_Hotspot,                        //mX4
21905: 875 uPSI_HList; //History List           //mX4
21906: 876 unit uPSI_DrTable;                   //VCL
21907: 877 uPSI_TConnect,                       //VCL
21908: 878 uPSI_DataBkr,                        //VCL
21909: 879 uPSI_HTTPIntr;                       //VCL
21910: 880 unit uPSI_Mathbox;                   //mX4
21911: 881 uPSI_cyIndy,                         //cY
21912: 882 uPSI_cySysUtils,                     //cY
21913: 883 uPSI_cyWinUtils,                     //cY
21914: 884 uPSI_cyStrUtils,                     //cY
21915: 885 uPSI_cyObjUtils,                     //cY
21916: 886 uPSI_cyDateUtils,                    //cY
21917: 887 uPSI_cyBDE,                          //cY
21918: 888 uPSI_cyClasses,                      //cY
21919: 889 uPSI_cyGraphics,  //3.9.9.94_2       //cY
21920: 890 unit uPSI_cyTypes;                   //cY
21921: 891 uPSI_JvDateTimePicker,               //JCL
21922: 892 uPSI_JvCreateProcess,                //JCL
21923: 893 uPSI_JvEasterEgg,                    //JCL
21924: 894 uPSI_WinSvc,  //3.9.9.94_3           //VCL
21925: 895 uPSI_SvcMgr                          //VCL
21926: 896 unit uPSI_JvPickDate;                //JCL
21927: 897 unit uPSI_JvNotify;                  //JCL
21928: 898 uPSI_JvStrHlder                      //JCL
21929: 899 unit uPSI_JclNTFS2;                  //JCL
21930: 900 uPSI_Jcl8087 //math coprocessor      //JCL
21931: 901 uPSI_JvAddPrinter                    //JCL
21932: 902 uPSI_JvCabFile                       //JCL
21933: 903 uPSI_JvDataEmbedded;                 //JCL
21934: 904 unit uPSI_U_HexView;                 //mX4
21935: 905 uPSI_UWavein4,                       //mX4
21936: 906 uPSI_AMixer,                         //mX4
21937: 907 uPSI_JvaScrollText,                  //mX4
21938: 908 uPSI_JvArrow,                        //mX4
21939: 909 unit uPSI_UrlMon;                    //mX4
21940: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'  //mX4
21941: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;  //DFF
21942: 912 unit uPSI_DFFUtils;                   //DFF
21943: 913 unit uPSI_MathsLib;                  //DFF
21944: 914 uPSI_UIntList;                       //DFF
21945: 915 uPSI_UGetParens;                     //DFF
21946: 916 unit uPSI_UGeometry;                 //DFF
21947: 917 unit uPSI_UAstronomy;                //DFF
21948: 918 unit uPSI_UCardComponentV2;          //DFF
21949: 919 unit uPSI_UTGraphSearch;             //DFF
21950: 920 unit uPSI_UParser10;                 //DFF
```

```
21951: 921 unit uPSI_cyIEUtils;                    //cY
21952: 922 unit uPSI_UcomboV2;                     //DFF
21953: 923 uPSI_cyBaseComm,                        //cY
21954: 924 uPSI_cyAppInstances,                    //cY
21955: 925 uPSI_cyAttract,                         //cY
21956: 926 uPSI_cyDERUtils                        //cY
21957: 927 unit uPSI_cyDocER;                      //cY
21958: 928 unit uPSI_ODBC;                         //mX
21959: 929 unit uPSI_AssocExec;                    //mX
21960: 930 uPSI_cyBaseCommRoomConnector,           //cY
21961: 931 uPSI_cyCommRoomConnector,               //cY
21962: 932 uPSI_cyCommunicate,                     //cY
21963: 933 uPSI_cyImage;                           //cY
21964: 934 uPSI_cyBaseContainer                    //cY
21965: 935 uPSI_cyModalContainer,                  //cY
21966: 936 uPSI_cyFlyingContainer;                 //cY
21967: 937 uPSI_RegStr,                           //VCL
21968: 938 uPSI_HtmlHelpViewer;                    //VCL
21969: 939 unit uPSI_cyIniForm                    //cY
21970: 940 unit uPSI_cyVirtualGrid;               //cY
21971: 941 uPSI_Profiler,                         //DA
21972: 942 uPSI_BackgroundWorker,                 //DA
21973: 943 uPSI_WavePlay,                         //DA
21974: 944 uPSI_WaveTimer,                        //DA
21975: 945 uPSI_WaveUtils;                        //DA
21976: 946 uPSI_NamedPipes,                       //TB
21977: 947 uPSI_NamedPipeServer,                  //TB
21978: 948 unit uPSI_process,                     //TB
21979: 949 unit uPSI_DPUtils;                     //TB
21980: 950 unit uPSI_CommonTools;                 //TB
21981: 951 uPSI_DataSendToWeb,                    //TB
21982: 952 uPSI_StarCalc,                         //TB
21983: 953 uPSI_D2_XPVistaHelperU                 //TB
21984: 954 unit uPSI_NetTools                     //TB
21985: 955 unit uPSI_Pipes                        //TB
21986: 956 uPSI_ProcessUnit,                      //mX
21987: 957 uPSI_adGSM,                            //mX
21988: 958 unit uPSI_BetterADODataSet;            //mX
21989: 959 unit uPSI_AdSelCom; //FTT             //mX
21990: 960 unit unit uPSI_dwsXPlatform;           //DWS
21991: 961 uPSI_AdSocket;                         //mX Turbo Power
21992: 962 uPSI_AdPacket;                         //mX
21993: 963 uPSI_AdPort;                           //mX
21994: 964 uPSI_PathFunc;                         //Inno
21995: 965 uPSI_CmnFunc;                          //Inno
21996: 966 uPSI_CmnFunc2; //Inno Setup            //Inno
21997: 967 unit uPSI_BitmapImage;                 //mX4
21998: 968 unit uPSI_ImageGrabber;                //mX4
21999: 969 uPSI_SecurityFunc,                     //Inno
22000: 970 uPSI_RedirFunc,                        //Inno
22001: 971 uPSI_FIFO, (MemoryStream)              //mX4
22002: 972 uPSI_Int64Em,                          //Inno
22003: 973 unit uPSI_InstFunc;                    //Inno
22004: 974 unit uPSI_LibFusion;                   //Inno
22005: 975 uPSI_SimpleExpression;                 //Inno
22006: 976 uPSI_unitResourceDetails,              //XN
22007: 977 uPSI_unitResFile,                      //XN
22008: 978 unit uPSI_simpleComport;               //mX4
22009: 979 unit uPSI_AfViewershelpers;            //Async
22010: 980 unit uPSI_Console;                     //mX4
22011: 981 unit uPSI_AnalogMeter;                 //TB
22012: 982 unit uPSI_XPrinter,                    //TB
22013: 983 unit uPSI_IniFiles;                    //VCL
22014: 984 unit uPSI_lazIniFiles;                 //FP
22015: 985 uPSI_testutils;                        //FP
22016: 986 uPSI_ToolsUnit; (DBTests)              //FP
22017: 987 uPSI_fpcunit                          //FP
22018: 988 uPSI_testdecorator;                    //FP
22019: 989 unit uPSI_fpcunittests;               //FP
22020: 990 unit uPSI_cTCPBuffer;                 //Fundamentals 4
22021: 991 unit uPSI_Glut,                       //Open GL
22022: 992 uPSI_LEDBitmaps,                      //mX4
22023: 993 uPSI_FileClass,                       //Inno
22024: 994 uPSI_FileUtilsClass,                  //mX4
22025: 995 uPSI_ComPortInterface; //Kit          //mX4
22026: 996 unit uPSI_SwitchLed;                  //mX4
22027: 997 unit uPSI_cyDmmCanvas;                //cY
22028: 998 uPSI_uColorFunctions;                 //DFF
22029: 999 uPSI_uSettings;                       //DFF
22030: 1000 uPSI_cyDebug.pas                      //cY
22031: 1001 uPSI_cyColorMatrix;                   //cY
22032: 1002 unit uPSI_cyCopyFiles;               //cY
22033: 1003 unit uPSI_cySearchFiles;              //cY
22034: 1004 unit uPSI_cyBaseMeasure;             //cY
22035: 1005 unit uPSI_PJIStreams;                //DD
22036: 1006 unit uPSI_cyRunTimeResize;           //cY
22037: 1007 unit uPSI_jcontrolutils;             //cY
22038: 1008 unit uPSI_kcMapViewer; (+GEONames)   //kc
22039: 1009 unit uPSI_kcMapViewerDESynapse;      //kc
```

```
22040: 1010 unit uPSI_cparserutils; (+GIS_SysUtils)   //kc
22041:
22042:
22043: ////////////////////////////////////////////////////////////////////////////
22044: //Form Template Library FTL
22045: ////////////////////////////////////////////////////////////////////////////
22046:
22047: 32 FTL For Form Building out of the Script, eg. 399_form_templates.txt
22048:
22049: 045 unit uPSI_VListView                      TFormListView;
22050: 263 unit uPSI_JvProfiler32;                  TProfReport
22051: 270 unit uPSI_ImgList;                       TCustomImageList
22052: 278 unit uPSI_JvImageWindow;                 TJvImageWindow
22053: 317 unit uPSI_JvParserForm;                  TJvHTMLParserForm
22054: 497 unit uPSI_DebugBox;                      TDebugBox
22055: 513 unit uPSI_ImageWin;                      TImageForm, TImageForm2
22056: 514 unit uPSI_CustomDrawTreeView;           TCustomDrawForm
22057: 515 unit uPSI_GraphWin;                     TGraphWinForm
22058: 516 unit uPSI_actionMain;                   TActionForm
22059: 518 unit uPSI_CtlPanel;                     TAppletApplication
22060: 529 unit uPSI_MDIEdit;                      TEditForm
22061: 535 unit uPSI_CoolMain; {browser}           TWebMainForm
22062: 538 unit uPSI_frmExportMain;                TSynexportForm
22063: 585 unit uPSI_usniffer; {//PortScanForm}    TSniffForm
22064: 600 unit uPSI_ThreadForm;                   TThreadSortForm;
22065: 618 unit uPSI_delphi_arduino_Unit1;         TLEDForm
22066: 620 unit uPSI_fplotMain;                    TfplotForm1
22067: 660 unit uPSI_JvDBQueryParamsForm;         TJvQueryParamsDialog
22068: 677 unit uPSI_OptionsFrm;                   TfrmOptions;
22069: 718 unit uPSI_MonForm;                     TMonitorForm
22070: 742 unit uPSI_SimplePortMain;              TPortForm1
22071: 770 unit uPSI_ovccalc;                     TOvcCalculator  //widget
22072: 810 unit uPSI_DbExcept;                    TDbEngineErrorDlg
22073: 812 unit uPSI_GL_actorUnit1;               TglActorForm1  //OpenGL Robot
22074: 846 unit uPSI_synhttp_daemon;              TTCPHttpDaemon, TTCPHttpThrd, TPingThread
22075: 867 unit uPSI_Debug;                       TfrmDebug
22076: 904 unit uPSI_U_HexView;                   THexForm2
22077: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)  TOscfrmMain
22078: 959 unit uPSI_AdSelCom;                    TComSelectForm
22079:
22080:
22081: ex.:with TEditForm.create(self) do begin
22082:       caption:= 'Template Form Tester';
22083:       FormStyle:= fsStayOnTop;
22084:       with editor do begin
22085:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
22086:         SelStart:= 0;
22087:         Modified:= False;
22088:       end;
22089:     end;
22090:   with TWebMainForm.create(self) do begin
22091:     URLs.Text:= 'http://www.kleiner.ch';
22092:     URLsClick(self); Show;
22093:   end;
22094:   with TSynexportForm.create(self) do begin
22095:     Caption:= 'Synexport HTML RTF tester';
22096:     Show;
22097:   end;
22098:   with TThreadSortForm.create(self) do begin
22099:      showmodal; free;
22100:   end;
22101:   with TCustomDrawForm.create(self) do begin
22102:        width:=820; height:=820;
22103:        image1.height:= 600; //add properties
22104:        image1.picture.bitmap:= image2.picture.bitmap;
22105:        //SelectionBackground1Click(self) CustomDraw1Click(self);
22106:        Background1.click;
22107:        bitmap1.click;
22108:        Tile1.click;
22109:        Showmodal;
22110:        Free;
22111:      end;
22112:    with TfplotForm1.Create(self) do begin
22113:      BtnPlotClick(self);
22114:      Showmodal; Free;
22115:    end;
22116:   with TOvcCalculator.create(self) do begin
22117:     parent:= aForm;
22118:    //free;
22119:     setbounds(550,510,200,150);
22120:     displaystr:= 'maXcalc';
22121:   end;
22122:   with THexForm2.Create(self) do begin
22123:      ShowModal;
22124:      Free;
22125:   end;
22126:
22127: function CheckBox: string;
22128: var idHTTP: TIDHTTP;
```

```
22129:  begin
22130:    result:= 'version not found';
22131:    if IsInternet then begin
22132:      idHTTP:= TIdHTTP.Create(NIL);
22133:      try
22134:        result:= idHTTP.Get(MXVERSIONFILE2);
22135:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
22136:        if result = MBVER2 then begin
22137:          //output.Font.Style:= [fsbold];
22138:          //Speak(' A new Version '+vstr+' of max box is available! ');
22139:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
22140:        end;
22141:       //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
22142:      finally
22143:        idHTTP.Free
22144:    end;
22145:  end;
22146: end;
22147:
22148:
22149: ////////////////////////////////////////////////////////////////////////////
22150: All maXbox Tutorials Table of Content 2014
22151: ////////////////////////////////////////////////////////////////////////////
22152:     Tutorial 00 Function-Coding  (Blix the Programmer)
22153:     Tutorial 01 Procedural-Coding
22154:     Tutorial 02 OO-Programming
22155:     Tutorial 03 Modular Coding
22156:     Tutorial 04 UML Use Case Coding
22157:     Tutorial 05 Internet Coding
22158:     Tutorial 06 Network Coding
22159:     Tutorial 07 Game Graphics Coding
22160:     Tutorial 08 Operating System Coding
22161:     Tutorial 09 Database Coding
22162:     Tutorial 10 Statistic Coding
22163:     Tutorial 11 Forms Coding
22164:     Tutorial 12 SQL DB Coding
22165:     Tutorial 13 Crypto Coding
22166:     Tutorial 14 Parallel Coding
22167:     Tutorial 15 Serial RS232 Coding
22168:     Tutorial 16 Event Driven Coding
22169:     Tutorial 17 Web Server Coding
22170:     Tutorial 18 Arduino System Coding
22171:     Tutorial 18_3 RGB LED System Coding
22172:     Tutorial 19 WinCOM /Arduino Coding
22173:     Tutorial 20 Regular Expressions RegEx
22174:     Tutorial 21 Android Coding (coming 2013)
22175:     Tutorial 22 Services Programming
22176:     Tutorial 23 Real Time Systems
22177:     Tutorial 24 Clean Code
22178:     Tutorial 25 maXbox Configuration I+II
22179:     Tutorial 26 Socket Programming with TCP
22180:     Tutorial 27 XML & TreeView
22181:     Tutorial 28 DLL Coding (available)
22182:     Tutorial 29 UML Scripting (2014)
22183:     Tutorial 30 Web of Things (2014)
22184:     Tutorial 31 Closures (2014)
22185:     Tutorial 32 SQL Firebird (coming 2014)
22186:     Tutorial 33 Oscilloscope (coming 2015)
22187:     Tutorial 34 GPS Navigation (2014)
22188:     Tutorial 35 Web Box (available)
22189:     Tutorial 36 Unit Testing (coming 2015)
22190:     Tutorial 37 API Coding (coming 2015)
22191:     Tutorial 38 3D Coding (coming 2015)
22192:     Tutorial 39 GEO Map Coding (coming 2015)
22193:
22194:
22195: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
22196: using Docu for this file is maxbox_functions_all.pdf
22197: PEP - Pascal Education Program  Lib Lab ShellHell
22198:
22199: http://stackoverflow.com/tags/pascalscript/hot
22200: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
22201: http://sourceforge.net/projects/maXbox   #locs:51620
22202: http://sourceforge.net/apps/mediawiki/maXbox
22203: http://www.blaisepascal.eu/
22204: https://github.com/maxkleiner/maXbox3.git
22205: http://www.heise.de/download/maxbox-1176464.html
22206: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
22207: https://www.facebook.com/pages/Programming-maXbox/166844836691703
22208: http://www.softwareschule.ch/arduino_training.pdf
22209: http://www.delphiarea.com
22210:
22211: All maXbox Examples List
22212: https://github.com/maxkleiner/maXbox3/releases
22213: ********************************************************************************************
22214: 000_pas_baseconvert.txt                          282_fadengraphik.txt
22215: 000_pas_baseconvert.txt_encrypt                  283_SQL_API_messagetimeout.txt
22216: 000_pas_baseconvert.txt_decrypt                  284_SysTools4.txt
22217: 001_1_pas_functest - Kopie.txt                  285_MineForm_GR32.TXT
```

22218: 001_1_pas_functest.txt
22219: 001_1_pas_functest2.txt
22220: 001_1_pas_functest_clx2.txt
22221: 001_1_pas_functest_clx2_2.txt
22222: 001_1_pas_functest_openarray.txt
22223: 001_pas_lottogen.txt
22224: 001_pas_lottogen_template.txt
22225: 001_pas_lottogen.txtcopy
22226: 002_pas_russianroulette.txt
22227: 002_pas_russianroulette.txtcopy
22228: 002_pas_russianroulette.txtcopy_decrypt
22229: 002_pas_russianroulette.txtcopy_encrypt
22230: 003_pas_motion.txt
22231: 003_pas_motion.txtcopy
22232: 004_pas_search.txt
22233: 004_pas_search_replace.txt
22234: 004_search_replace_allfunctionlist.txt
22235: 005_pas_oodesign.txt
22236: 005_pas_shelllink.txt
22237: 006_pas_oobatch.txt
22238: 007_pas_streamcopy.txt
22239: 008_EINMALEINS_FUNC.TXT
22240: 008_explanation.txt
22241: 008_pas_verwechselt.txt
22242: 008_pas_verwechselt_ibz_bern_func.txt
22243: 008_stack_ibz.TXT
22244: 009_pas_umlrunner.txt
22245: 009_pas_umlrunner_all.txt
22246: 009_pas_umlrunner_componenttest.txt
22247: 009_pas_umlrunner_solution.txt
22248: 009_pas_umlrunner_solution_2step.txt
22249: 010_pas_oodesign_solution.txt
22250: 011_pas_puzzlepas_defect.txt
22251: 012_pas_umlrunner_solution.txt
22252: 012_pas_umlrunner_solution2.txt
22253: 013_pas_linenumber.txt
22254: 014_pas_primetest.txt
22255: 014_pas_primetest_first.txt
22256: 014_pas_primetest_sync.txt
22257: 015_pas_designbycontract.txt
22258: 015_pas_designbycontract_solution.txt
22259: 016_pas_searchrec.txt
22260: 017_chartgen.txt
22261: 018_data_simulator.txt
22262: 019_dez_to_bin.txt
22263: 019_dez_to_bin_grenzwert_ibz.txt
22264: 020_proc_feedback.txt
22265: 021_pas_symkey.txt
22266: 021_pas_symkey_solution.txt
22267: 022_pas_filestreams.txt
22268: 023_pas_find_searchrec.txt
22269: 023_pas_pathfind.txt
22270: 024_pas_TFileStream_records.txt
22271: 025_prime_direct.txt
22272: 026_pas_memorystream.txt
22273: 027_pas_shellexecute_beta.txt
22274: 027_pas_shellexecute_solution.txt
22275: 028_pas_dataset.txt
22276: 029_pas_assignfile.txt
22277: 029_pas_assignfile_dragndropexe.txt
22278: 030_palindrome_2.txt
22279: 030_palindrome_tester.txt
22280: 030_pas_recursion.txt
22281: 030_pas_recursion2.txt
22282: 031_pas_hashcode.txt
22283: 032_pas_crc_const.txt
22284: 033_pas_cipher.txt
22285: 033_pas_cipher_def.txt
22286: 033_pas_cipher_file_2_solution.txt
22287: 034_pas_soundbox.txt
22288: 035_pas_crcscript.txt
22289: 035_pas_CRCscript_modbus.txt
22290: 036_pas_includetest.txt
22291: 036_pas_includetest_basta.txt
22292: 037_pas_define_demo32.txt
22293: 038_pas_box_demonstrator.txt
22294: 039_pas_dllcall.txt
22295: 040_paspointer.txt
22296: 040_paspointer_old.txt
22297: 041_pasplotter.txt
22298: 041_pasplotter_plus.txt
22299: 042_pas_kgv_ggt.txt
22300: 043_pas_proceduretype.txt
22301: 044_pas_14queens_solwith14.txt
22302: 044_pas_8queens.txt
22303: 044_pas_8queens_sol2.txt
22304: 044_pas_8queens_solutions.txt
22305: 044_queens_performer.txt
22306: 044_queens_performer2.txt
285_MineForm_GR32main.TXT
285_MineForm_GR32mainsolution.TXT
285_MineForm_propas.TXT
285_MineForm_propas2.TXT
285_minesweeper2.TXT
285_Patterns_process.txt
286_colormixer_jpeg_charcounter.txt
286_colormixer_jpeg_charcounter2.txt
287_eventhandling.txt
287_eventhandling2.txt
287_eventhandling2_negpower.txt
288_bitblt.txt
288_bitblt_resize.txt
289_regression.txt
289_regression2.txt
290_bestofbox.txt
290_bestofbox2.txt
290_bestofbox3.txt
291_3sort_visual_thread.txt
292_refactoring2.txt
293_bold_utils.txt
293_ib_utils.txt
293_ib_utils_timetest.txt
294_maxcalc_demo.txt
294_maxcalc_demo2.txt
295_easter_calendar.txt
295_easter_calendar2.txt
295_easter_combobox.txt
297_atomimage.txt
297_atomimage2.txt
297_atomimage3.txt
297_atomimage4.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt
299_animation.txt
299_animationmotor_arduino.txt
299_animation_formprototype.txt
299_realtimeclock_arduino.txt
299_realtimeclock_arduino2.txt
300_treeview.txt
300_treeview_test.txt
300_treeview_test2.txt
300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
309_regex_powertester3.txt
310_regex_decorator.TXT
312_ListView.txt
313_dmath_dll.txt
314_fundamentals4_tester.TXT

22307: 044_queens_performer2tester.txt
22308: 045_pas_listhandling.txt
22309: 046_pas_records.txt
22310: 047_pas_modula10.txt
22311: 048_pas_romans.txt
22312: 049_pas_ifdemo.txt
22313: 049_pas_ifdemo_BROKER.txt
22314: 050_pas_primetest2.txt
22315: 050_pas_primetester_thieves.txt
22316: 050_program_starter.txt
22317: 050_program_starter_performance.txt
22318: 051_pas_findtext_solution.txt
22319: 052_pas_text_as_stream.txt
22320: 052_pas_text_as_stream_include.txt
22321: 053_pas_singleton.txt
22322: 054_pas_speakpassword.txt
22323: 054_pas_speakpassword2.txt
22324: 054_pas_speakpassword_searchtest.txt
22325: 055_pas_factorylist.txt
22326: 056_pas_demeter.txt
22327: 057_pas_dirfinder.txt
22328: 058_pas_filefinder.txt
22329: 058_pas_filefinder_pdf.txt
22330: 058_pas_filefinder_screview.txt
22331: 058_pas_filefinder_screview2.txt
22332: 058_pas_filefinder_screview3.txt
22333: 059_pas_timertest.txt
22334: 059_pas_timertest_2.txt
22335: 059_pas_timertest_time_solution.txt
22336: 059_timerobject_starter2.txt
22337: 059_timerobject_starter2_ibz2_async.txt
22338: 059_timerobject_starter2_uml.txt
22339: 059_timerobject_starter2_uml_main.txt
22340: 059_timerobject_starter4_ibz.txt
22341: 060_pas_datefind.txt
22342: 060_pas_datefind_exceptions2.txt
22343: 060_pas_datefind_exceptions_CHECKTEST.txt
22344: 060_pas_datefind_fulltext.txt
22345: 060_pas_datefind_plus.txt
22346: 060_pas_datefind_plus_mydate.txt
22347: 061_pas_randomwalk.txt
22348: 061_pas_randomwalk_plus.txt
22349: 062_paskorrelation.txt
22350: 063_pas_calculateform.txt
22351: 063_pas_calculateform_2list.txt
22352: 064_pas_timetest.txt
22353: 065_pas_bitcounter.txt
22354: 066_pas_eliza.txt
22355: 066_pas_eliza_include_sol.txt
22356: 067_pas_morse.txt
22357: 068_pas_piezo_sound.txt
22358: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
22359: 069_my_LEDBOX.TXT
22360: 069_pas_ledmatrix.txt
22361: 069_pas_LEDMATRIX_Alphabet.txt
22362: 069_pas_LEDMATRIX_Alphabet_run.txt
22363: 069_pas_LEDMATRIX_Alphabet_tester.txt
22364: 069_PAS_LEDMATRIX_COLOR.TXT
22365: 069_pas_ledmatrix_fixedit.txt
22366: 069_pas_LEDMATRIX_soundbox.txt
22367: 069_pas_LEDMATRIX_soundbox2.txt
22368: 069_Richter_MATRIX.TXT
22369: 070_pas_functionplot.txt
22370: 070_pas_functionplotter2.txt
22371: 070_pas_functionplotter2_mx4.txt
22372: 070_pas_functionplotter2_tester.txt
22373: 070_pas_functionplotter3.txt
22374: 070_pas_functionplotter4.txt
22375: 070_pas_functionplotter_digital.txt
22376: 070_pas_functionplotter_elliptic.txt
22377: 070_pas_function_helmholtz.txt
22378: 070_pas_textcheck_experimental.txt
22379: 071_pas_graphics.txt
22380: 071_pas_graphics_drawsym.txt
22381: 071_pas_graphics_drawsym_save.txt
22382: 071_pas_graphics_random.txt
22383: 072_pas_fractals.txt
22384: 072_pas_fractals_2.txt
22385: 072_pas_fractals_blackhole.txt
22386: 072_pas_fractals_perfomance.txt
22387: 072_pas_fractals_perfomance_new.txt
22388: 072_pas_fractals_perfomance_sharp.txt
22389: 072_pas_fractals_performance.txt
22390: 072_pas_fractals_performance_mX4.txt
22391: 073_pas_forms.txt
22392: 074_pas_chartgenerator.txt
22393: 074_pas_chartgenerator_solution.txt
22394: 074_pas_chartgenerator_solution_back.txt
22395: 074_pas_charts.txt
315_funcplot_dmath.TXT
316_cfileutils_cdatetime_tester.TXT
317_excel_export_tester.TXT
318_excel_export.TXT
318_excel_export2.TXT
318_excel_export3.TXT
318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
330_myclock.txt
330_myclock2.txt
331_SQL_DBfirebird4.txt
332_jprofiler.txt
332_jprofiler_form.txt
332_jprofiler_form2.txt
333_querybyexample.txt
333_querybyexample2.txt
334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docutype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_duallistview.txt
349_biginteger.txt
350_parserform.txt
351_chartform.txt
351_chartform2.txt
351_chartform3.txt
352_array_unittest.txt
353_smtp_email.txt
353_smtp_email2.txt
354_josephus.txt
355_life_of_PI.txt
356_3D_printer.txt
357_fplot.TXT
358_makesound.txt
359_charsetrules.TXT
360_allobjects.TXT
360_JvPaintFX.TXT
361_heartbeat_wave.TXT
362_maxonmotor2.TXT
363_compress_services.txt
363_compress_services2.txt
364_pdf_services.txt

22396: 075_bitmap_Artwork2.txt
22397: 075_pas_bitmappuzzle.txt
22398: 075_pas_bitmappuzzle24.prod.txt
22399: 075_pas_bitmappuzzle2_prod.txt
22400: 075_pas_bitmappuzzle3.txt
22401: 075_pas_bitmapsolve.txt
22402: 075_pas_bitmap_Artwork.txt
22403: 075_pas_puzzlepas_solution.txt
22404: 076_pas_3dcube.txt
22405: 076_pas_circle.txt
22406: 077_pas_mmshow.txt
22407: 078_pas_pi.txt
22408: 079_pas_3dcube_animation.txt
22409: 079_pas_3dcube_animation4.txt
22410: 079_pas_3dcube_plus.txt
22411: 080_pas_hanoi.txt
22412: 080_pas_hanoi2.txt
22413: 080_pas_hanoi2_file.txt
22414: 080_pas_hanoi2_sol.txt
22415: 080_pas_hanoi2_tester.txt
22416: 080_pas_hanoi2_tester_fast.txt
22417: 080_pas_hanoi3.txt
22418: 081_pas_chartist2.txt
22419: 082_pas_biorythmus.txt
22420: 082_pas_biorythmus_solution.txt
22421: 082_pas_biorythmus_solution_3.txt
22422: 082_pas_biorythmus_test.txt
22423: 083_pas_GITARRE.txt
22424: 083_pas_soundbox_tones.txt
22425: 084_pas_waves.txt
22426: 085_mxsinus_logo.txt
22427: 085_sinus_plot_waves.txt
22428: 086_pas_graph_arrow_heart.txt
22429: 087_bitmap_loader.txt
22430: 087_pas_bitmap_solution.txt
22431: 087_pas_bitmap_solution2.txt
22432: 087_pas_bitmap_subimage.txt
22433: 087_pas_bitmap_test.txt
22434: 088_pas_soundbox2_mp3.txt
22435: 088_pas_soundbox_mp3.txt
22436: 088_pas_sphere_2.txt
22437: 089_pas_gradient.txt
22438: 089_pas_maxland2.txt
22439: 090_pas_sudoku4.txt
22440: 090_pas_sudoku4_2.txt
22441: 091_pas_cube4.txt
22442: 092_pas_statistics4.txt
22443: 093_variance.txt
22444: 093_variance_debug.txt
22445: 094_pas_daysold.txt
22446: 094_pas_stat_date.txt
22447: 095_pas_ki_simulation.txt
22448: 096_pas_geisen_problem.txt
22449: 096_pas_montyhall_problem.txt
22450: 097_lotto_proofofconcept.txt
22451: 097_pas_lottocombinations_beat_plus.txt
22452: 097_pas_lottocombinations_beat_plus2.txt
22453: 097_pas_lottocombinations_universal.txt
22454: 097_pas_lottosimulation.txt
22455: 098_pas_chartgenerator_plus.txt
22456: 099_pas_3D_show.txt
22457: 200_big_numbers.txt
22458: 200_big_numbers2.txt
22459: 201_streamload_xml.txt
22460: 202_systemcheck.txt
22461: 203_webservice_simple_intftester.txt
22462: 204_webservice_simple.txt
22463: 205_future_value_service.txt
22464: 206_DTD_string_functions.txt
22465: 207_ibz2_async_process.txt
22466: 208_crc32_hash.txt
22467: 209_cryptohash.txt
22468: 210_public_private.txt
22469: 210_public_private_cryptosystem.txt
22470: 211_wipe_pattern.txt
22471: 211_wipe_pattern2.txt
22472: 211_wipe_pattern_solution.txt
22473: 212_pas_statisticmodule4.TXT
22474: 212_pas_statisticmoduletxt.TXT
22475: 212_statisticmodule4.txt
22476: 213_pas_BBP_Algo.txt
22477: 214_mxdocudemo.txt
22478: 214_mxdocudemo2.txt
22479: 214_mxdocudemo3.txt
22480: 215_hints_test.TXT
22481: 216_warnings_test.TXT
22482: 217_pas_heartbeat.txt
22483: 218_biorythmus_studio.txt
22484: 219_cipherbox.txt

365_memorystream.txt
365_memorystream2.txt
365_memorystream_test.txt
365_U_HexView.txt
366_mp3player.txt
366_mp3player2.txt
366_mp3player2_themestest.txt
367_silvi_player_widgets.txt
367_silvi_player_widgets2.txt
367_widgets.txt
368_configuration_demo.txt
369_macro_demo.txt
370_callback2grid.TXT
370_richedit.txt
370_richedit_highlight.txt
370_synedit.txt
370_synedit2.txt
370_synedit2_mxtester.txt
370_synedit2_mxtester2.txt
371_maXbook_v4tester.txt
372stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
377_smartXMLWorkshop.txt
377_smartXMLWorkshop2.txt
378_queryperformance3.txt
378_REST1.txt
378_REST2.txt
379_timefunc.txt
379_timefunctesterfilemon.txt
380_coolfunc.txt
380_coolfunc2.txt
380_coolfunc_tester.txt
381_bitcoin_simulation.txt
382_GRMath.TXT
382_GRMath_PI_Proof.TXT
382_GRMath_Riemann.TXT
383_MDAC_DCOM.txt
384_TeamViewerID.TXT
386_InternetRadio.TXT
387_fulltextfinder.txt
387_fulltextfinder_cleancode.txt
387_fulltextfinder_fast.txt
387_fulltext_getscripttest.txt
388_TCPServerSock.TXT
388_TCPServerSock2.TXT
388_TCPServerSockClient.TXT
389_TAR_Archive.TXT
389_TAR_Archive_test.TXT
389_TAR_Archive_test2.TXT
390_Callback3.TXT
390_Callback3Rec.TXT
390_CallbackClean.TXT
390_StringlistHTML.TXT
391_ToDo_List.TXT
392_Barcode.TXT
392_Barcode2.TXT
392_Barcode23.TXT
392_Barcode2scholz.TXT
392_Barcode3scholz.TXT
393_QRCode.TXT
393_QRCode2.TXT
393_QRCode2Direct.TXT
393_QRCode2DirectIndy.TXT
393_QRCode2Direct_detlef.TXT
393_QRCode3.TXT
394_networkgraph.TXT
394_networkgraph_depwalkutilstest.TXT
394_networkgraph_depwalkutilstest2.TXT
395_USBController.TXT
396_Sort.TXT
397_Hotlog.TXT
397_Hotlog2.TXT
398_ustrings.txt
399_form_templates.txt
400_fplottchart.TXT
400_fplottchart2.TXT
400_fplottchart2teetest.TXT
400_QRCodeMarket.TXT
401_tfilerun.txt

22485: 219_crypt_source_comtest_solution.TXT
22486: 220_cipherbox_form.txt
22487: 220_cipherbox_form2.txt
22488: 221_bcd_explain.txt
22489: 222_memoform.txt
22490: 223_directorybox.txt
22491: 224_dialogs.txt
22492: 225_sprite_animation.txt
22493: 226_ASCII_Grid2.TXT
22494: 227_animation.txt
22495: 227_animation2.txt
22496: 228_android_calendar.txt
22497: 229_android_game.txt
22498: 229_android_game_tester.txt
22499: 230_DataProvider.txt
22500: 230_DataSetProvider.txt
22501: 230_DataSetXMLBackupScholz.txt
22502: 231_DBGrid_access.txt
22503: 231_DBGrid_XMLaccess.txt
22504: 231_DBGrid_XMLaccess2.txt
22505: 231_DBGrid_XMLaccess_locatetester.txt
22506: 231_DBGrid_XML_CDS_local.txt
22507: 232_outline.txt
22508: 232_outline_2.txt
22509: 233_modular_form.txt
22510: 234_debugoutform.txt
22511: 235_fastform.TXT
22512: 236_componentpower.txt
22513: 236_componentpower_back.txt
22514: 237_pas_4forms.txt
22515: 238_lottogen_form.txt
22516: 239_pas_sierpinski.txt
22517: 239_pas_sierpinski2.txt
22518: 240_unitGlobal_tester.txt
22519: 241_db3_sql_tutorial.txt
22520: 241_db3_sql_tutorial2.txt
22521: 241_db3_sql_tutorial2fix.txt
22522: 241_db3_sql_tutorial3.txt
22523: 241_db3_sql_tutorial3connect.txt
22524: 241_db3_sql_tutorial3_fptest.txt
22525: 241_RTL_SET2.txt
22526: 241_RTL_SET2_tester.txt
22527: 242_Component_Control.txt
22528: 243_tutorial_loader.txt
22529: 244_script_loader_loop.txt
22530: 245_formapp2.txt
22531: 245_formapp2_tester.txt
22532: 245_formapp2_testerX.txt
22533: 246_httpapp.txt
22534: 247_datecalendar.txt
22535: 248_ASCII_Grid2_sorted.TXT
22536: 249_picture_grid.TXT
22537: 250_tipsandtricks2.txt
22538: 250_tipsandtricks3.txt
22539: 250_tipsandtricks3api.txt
22540: 250_tipsandtricks3_admin_elevation.txt
22541: 250_tipsandtricks3_tester.txt
22542: 250_tipsandtricks4_tester.txt
22543: 250_tipsandtricks4_tester2.txt
22544: 251_compare_noise_gauss.txt
22545: 251_whitenoise.txt
22546: 251_whitenoise2.txt
22547: 252_hilbert_turtle.txt
22548: 252_pas_hilbert.txt
22549: 253_opearatingsystem3.txt
22550: 254_dynarrays.txt
22551: 255_einstein.txt
22552: 256_findconsts_of_EXE.txt
22553: 256_findfunctions2_of_EXE.txt
22554: 256_findfunctions2_of_EXEaverp.txt
22555: 256_findfunctions2_of_EXEspec.txt
22556: 256_findfunctions3.txt
22557: 256_findfunctions_of_EXE.txt
22558: 257_AES_Cipher.txt
22559: 258_AES_cryptobox.txt
22560: 258_AES_cryptobox2.txt
22561: 258_AES_cryptobox2_passdlg.txt
22562: 259_AES_crypt_directory.txt
22563: 260_sendmessage_2.TXT
22564: 260_sendmessage_beta.TXT
22565: 261_probability.txt
22566: 262_mxoutputdemo4.txt
22567: 263_async_sound.txt
22568: 264_vclutils.txt
22569: 264_VCL_utils2.txt
22570: 265_timer_API.txt
22571: 266_serial_interface.txt
22572: 266_serial_interface2.txt
22573: 266_serial_interface3.txt

402_richedit2.txt
403_outlookspy.txt
404_simplebrowser.txt
405_datefinder_today.txt
406_portscan.txt
407_indydemo.txt
408_testroboter.txt
409_excel_control.txt
410_keyboardevent.txt
411_json_test.txt
412_Zeosutils.txt
413_listview2.txt
414_avrdude_flash.txt
415_avrdude_writehex.txt
416_sonar_startscriptEKON.TXT
416_sonar_startscriptEKON_reporting.TXT
417_GRMath_PI_Proof2.TXT
418_functional_paradigm.txt
419_archimedes_spiral.txt
419_archimedes_spiral2.txt
420_archimedes_arduino.txt
420_Lissajous.txt
421_PI_Power.TXT
421_PI_Power2.TXT
422_world_bitboxx.txt
423_game_of_life.TXT
423_game_of_life2.TXT
423_game_of_life3.TXT
423_game_of_life3_test.TXT
423_game_of_life4.TXT
423_game_of_life4_kryptum.TXT
424_opengl_tester.txt
425_reversi_game.txt
426_IBUtils.TXT
427_IBDatabase.TXT
428_SortGrid.TXT
429_fileclass.txt
430_fileoperation.txt
430_fileoperation_tester.txt
431_performance_index.txt
432_shortstring_routines.txt
433_video_avicap.txt
433_video_avicap2.txt
434_GSM_module.TXT
435_httpcommon.txt
436_GraphicSplitter.txt
436_GraphicSplitter_form.txt
436_GraphicSplitter_form2.txt
436_teetest_screen.TXT
436_teetest_screen2.TXT
437_WinAPItop.txt
437_WinAPItop_Firebirdtester.txt
438_OvcInternational.txt
439_AsyncFreeDemo.txt
439_AsyncFreeDemoForm.txt
440_DLL_Tutor.txt
440_DLL_Tutor2.txt
440_XML_Tutor.txt
440_XML_Tutor2.txt
441_make_app.txt
442_arduino_rgb_led.txt
443_webserver_arduino_rgb_light.txt
443_webserver_arduino_rgb_light4.txt
444_webserver_arduino3ibz_rgb_led_basta.txt
445_datagrid.txt
445_datagrid2.txt
445_datagrid_android_arduino.txt
446_arduino_timer.txt
447_patternFrm_mx3.txt
448_Synapse.txt
448_Synapse2.txt
449_dweb_start_tester.txt
450_Synapse_HTTPS.txt
450_Synapse_Mime.txt
450_Synapse_ScanPing.txt
451_ocx_player.txt
451_OCX_WinPlayer2.txt
452_dbtreeview.txt
452_dbtreeview2.txt
453_stdfuncs.txt
454_fileStream.txt
455_functionfun.txt
455_functionfun2.txt
455_functionfun2_test.txt
457_ressource_grid.txt
458_atomimageX.txt
459_cindyfunc.txt
459_cindyfunc2.txt
460_TopTenFunctions.txt

```
22574: 267_ackermann_rec.txt                          461_sqlform_calwin.txt
22575: 267_ackermann_variants.txt                     462_caesarcipher.txt
22576: 268_DBGrid_tree.txt                            463_global_exception.txt
22577: 269_record_grid.TXT                            464_function_procedure.txt
22578: 270_Jedi_FunctionPower.txt                     464_function_procedure2.txt
22579: 270_Jedi_FunctionPowertester.txt               464_function_procedure3.txt
22580: 271_closures_study.txt                         465_U_HexView.txt
22581: 271_closures_study_workingset2.txt             466_moon.txt
22582: 272_pas_function_show.txt                      466_moon_inputquery.txt
22583: 273_pas_function_show2.txt                     4671_cardmagic.txt
22584: 274_library_functions.txt                      467_helmholtz_graphic.txt
22585: 275_turtle_language.txt                        468_URLMon.txt
22586: 275_turtle_language_save.txt                   468_URLMon2.txt
22587: 276_save_algo.txt                             469_formarrow.txt
22588: 276_save_algo2.txt                            469_formarrow_datepicker.txt
22589: 277_functionsfor39.txt                        469_formarrow_datepicker_ibz_result.txt
22590: 278_DB_Dialogs.TXT                            469_ibzresult.txt
22591: 279_hexer2.TXT                                470_DFFUtils_compiled.txt
22592: 279_hexer2macro.TXT                           470_DFFUtils_ScrollingLED.txt
22593: 279_hexer2macroback.TXT                       470_Oscilloscope.txt
22594: 280_UML_process.txt                           470_Oscilloscope_code.txt
22595: 280_UML_process_knabe2.txt                    471_cardmagic.txt
22596: 280_UML_process_knabe3.txt                    471_cardmagic2.txt
22597: 280_UML_process_TIM_Botzenhardt.txt           472_allcards.TXT
22598: 280_UML_TIM_Seitz.txt                         473_comboset.txt
22599: 281_picturepuzzle.txt                         474_wakeonlan.txt
22600: 281_picturepuzzle2.txt                        474_wakeonlan2.txt
22601: 281_picturepuzzle3.txt                        476_getscripttest.txt
22602: 281_picturepuzzle4.txt                        477_filenameonly.txt
22603: 479_inputquery.txt                            480_regex_pathfinder.txt
22604: 480_regex_pathfinder2.txt                     481_processList.txt
22605: 482 processPipe.txt                           482 processPipeGCC.txt
22606: 483_PathFuncTest_mX.txt                       484_filefinder3.txt
22607: 485_InnoFunc.txt                              486_VideoGrabber.txt
22608: 487_asyncKeyState.txt                         488_asyncTerminal.txt
22609: 489_simpleComport.txt                         490_webCamproc.txt
22610: 491_analogmeter.txt                           492_snowflake2.txt
22611: 493_gadgets.txt                               495_fourierfreq.txt
22612: 496_InstallX.txt                              497_LED.txt
22613: 498_UnitTesting.txt                           499_mulu42.txt
22614: 500_diceoflifes.txt                           501_firebird_datasnap_tests.txt
22615: 502_findalldocs.txt                           503_led_switch.txt
22616: 504_fileclass.txt                             505_debug.txt
22617: 506_colormatrix.txt                           507_derutils.txt
22618: 508_simplecomportmorse.txt                    509_GEOMap2.txt
22619:
22620:
22621: Web Script Examples:
22622:
22623: http://www.softwareschule.ch/examples/performer.txt';
22624: http://www.softwareschule.ch/examples/turtle.txt';
22625: http://www.softwareschule.ch/examples/SQLExport.txt';
22626: http://www.softwareschule.ch/examples/Richter.txt';
22627: http://www.softwareschule.ch/examples/checker.txt';
22628: http://www.softwareschule.ch/examples/demoscript.txt';
22629: http://www.softwareschule.ch/examples/ibzresult.txt';
22630: http://www.softwareschule.ch/examples/performindex.txt
22631: http://www.softwareschule.ch/examples/processlist.txt
22632: http://www.softwareschule.ch/examples/game.txt
22633:
22634:
22635:
22636: Delphi Basics Run Time Library listing
22637: ********************************************************************************
22638: A
22639: Compiler Directive $A Determines whether data is aligned or packed
22640: Compiler Directive $Align Determines whether data is aligned or packed
22641: Compiler Directive $AppType Determines the application type : GUI or Console
22642: Procedure SysUtils Abort Aborts the current processing with a silent exception
22643: Function System Abs Gives the absolute value of a number (-ve sign is removed)
22644: Directive Abstract Defines a class method only implemented in subclasses
22645: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
22646: Function System Addr Gives the address of a variable, function or procedure
22647: Keyword And Boolean and or bitwise and of two arguments
22648: Type System AnsiChar A character type guaranteed to be 8 bits in size
22649: Function SysUtils AnsiCompareStr Compare two strings for equality
22650: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
22651: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
22652: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
22653: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
22654: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
22655: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
22656: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
22657: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
22658: Function StrUtils AnsiPos Find the position of one string in another
22659: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
22660: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
22661: Function StrUtils AnsiRightStr Extracts characters from the right of a string
22662: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
```

```
22663: Type System AnsiString A data type that holds a string of AnsiChars
22664: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
22665: Procedure System Append Open a text file to allow appending of text to the end
22666: Procedure SysUtils AppendStr Concatenate one string onto the end of another
22667: Function Math ArcCos The Arc Cosine of a number, returned in radians
22668: Function Math ArcSin The Arc Sine of a number, returned in radians
22669: Function System ArcTan The Arc Tangent of a number, returned in radians
22670: Keyword Array A data type holding indexable collections of data
22671: Keyword As Used for casting object references
22672: Procedure System Assign Assigns a file handle to a binary or text file
22673: Function System Assigned Returns true if a reference is not nil
22674: Procedure System AssignFile Assigns a file handle to a binary or text file
22675: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
22676:
22677: B
22678: Compiler Directive $B Whether to short cut and and or operations
22679: Compiler Directive $BoolEval Whether to short cut and and or operations
22680: Procedure SysUtils Beep Make a beep sound
22681: Keyword Begin Keyword that starts a statement block
22682: Function System BeginThread Begins a separate thread of code execution
22683: Procedure System BlockRead Reads a block of data records from an untyped binary file
22684: Procedure System BlockWrite Writes a block of data records to an untyped binary file
22685: Type System Boolean Allows just True and False values
22686: Function Classes Bounds Create a TRect value from top left and size values
22687: Procedure System Break Forces a jump out of a single loop
22688: Type System Byte An integer type supporting values 0 to 255
22689:
22690: C
22691: Type System Cardinal The basic unsigned integer type
22692: Keyword Case A mechanism for acting upon different values of an Ordinal
22693: Function StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
22694: Function SysUtils ChangeFileExt Change the extension part of a file name
22695: Type System Char Variable type holding a single character
22696: Procedure System ChDir Change the working drive plus path for a specified drive
22697: Function System Chr Convert an integer into a character
22698: Keyword Class Starts the declaration of a type of object class
22699: Procedure System Close Closes an open file
22700: Procedure System CloseFile Closes an open file
22701: Variable System CmdLine Holds the execution text used to start the current program
22702: Type System Comp A 64 bit signed integer
22703: Function SysUtils CompareStr Compare two strings to see which is greater than the other
22704: Function SysUtils CompareText Compare two strings for equality, ignoring case
22705: Function Math CompareValue Compare numeric values with a tolerance
22706: Function System Concat Concatenates one or more strings into one string
22707: Keyword Const Starts the definition of fixed data values
22708: Keyword Constructor Defines the method used to create an object from a class
22709: Procedure System Continue Forces a jump to the next iteration of a loop
22710: Function ConvUtils Convert Convert one measurement value to another
22711: Function System Copy Create a copy of part of a string or an array
22712: Function System Cos The Cosine of a number
22713: Function SysUtils CreateDir Create a directory
22714: Type System Currency A floating point type with 4 decimals used for financial values
22715: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
22716: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
22717: Variable SysUtils CurrencyString The currency string used in currency display functions
22718: Function SysUtils CurrToStr Convert a currency value to a string
22719: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
22720:
22721: D
22722: Compiler Directive $D Determines whether application debug information is built
22723: Compiler Directive $DebugInfo Determines whether application debug information is built
22724: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
22725: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
22726: Function SysUtils Date Gives the current date
22727: Variable SysUtils DateSeparator The character used to separate display date fields
22728: Function SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
22729: Function SysUtils DateTimeToStr Converts TDateTime date and time values to a string
22730: Procedure SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
22731: Function SysUtils DateToStr Converts a TDateTime date value to a string
22732: Function DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
22733: Function DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
22734: Function DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
22735: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
22736: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
22737: Function DateUtils DaysInAMonth Gives the number of days in a month
22738: Function DateUtils DaysInAYear Gives the number of days in a year
22739: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
22740: Procedure System Dec Decrement an ordinal variable
22741: Variable SysUtils DecimalSeparator The character used to display the decimal point
22742: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
22743: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
22744: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
22745: Directive Default Defines default processing for a property
22746: Function Math DegToRad Convert a degrees value to radians
22747: Procedure System Delete Delete a section of characters from a string
22748: Function SysUtils DeleteFile Delete a file specified by its file name
22749: Keyword Destructor Defines the method used to destroy an object
22750: Function SysUtils DirectoryExists Returns true if the given directory exists
22751: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
```

```
22752: Function SysUtils DiskSize Gives the size in bytes of a specified drive
22753: Procedure System Dispose Dispose of storage used by a pointer type variable
22754: Keyword Div Performs integer division, discarding the remainder
22755: Keyword Do Defines the start of some controlled action
22756: Type System Double A floating point type supporting about 15 digits of precision
22757: Keyword DownTo Prefixes an decremental for loop target value
22758: Function StrUtils DupeString Creates a string containing copies of a substring
22759: Directive Dynamic Allows a class method to be overriden in derived classes
22760:
22761: E
22762: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
22763: Compiler Directive $EndIf Terminates conditional code compilation
22764: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
22765: Keyword Else Starts false section of if, case and try statements
22766: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
22767: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
22768: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
22769: Keyword End Keyword that terminates statement blocks
22770: Function DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
22771: Function DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
22772: Procedure System EndThread Terminates a thread with an exit code
22773: Function System Eof Returns true if a file opened with Reset is at the end
22774: Function System Eoln Returns true if the current text file is pointing at a line end
22775: Procedure System Erase Erase a file
22776: Variable System ErrorAddr Sets the error address when an application terminates
22777: Keyword Except Starts the error trapping clause of a Try statement
22778: Procedure System Exclude Exclude a value in a set variable
22779: Procedure System Exit Exit abruptly from a function or procedure
22780: Variable System ExitCode Sets the return code when an application terminates
22781: Function System Exp Gives the exponent of a number
22782: Directive System Export Makes a function or procedure in a DLL externally available
22783: Type System Extended The floating point type with the highest capacity and precision
22784: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
22785: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
22786: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
22787: Function SysUtils ExtractFileName Extracts the name part of a full file name
22788: Function SysUtils ExtractFilePath Extracts the path part of a full file name
22789:
22790: F
22791: Function StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
22792: Keyword File Defines a typed or untyped file
22793: Function SysUtils FileAge Get the last modified date/time of a file without opening it
22794: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
22795: Function SysUtils FileExists Returns true if the given file exists
22796: Function SysUtils FileGetAttr Gets the attributes of a file
22797: Variable System FileMode Defines how Reset opens a binary file
22798: Function System FilePos Gives the file position in a binary or text file
22799: Function SysUtils FileSearch Search for a file in one or more directories
22800: Function SysUtils FileSetAttr Sets the attributes of a file
22801: Function SysUtils FileSetDate Set the last modified date and time of a file
22802: Function System FileSize Gives the size in records of an open file
22803: Procedure System FillChar Fills out a section of storage with a fill character or byte value
22804: Keyword Finally Starts the unconditional code section of a Try statement
22805: Function SysUtils FindClose Closes a successful FindFirst file search
22806: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
22807: Function SysUtils FindFirst Finds all files matching a file mask and attributes
22808: Function SysUtils FindNext Find the next file after a successful FindFirst
22809: Function SysUtils FloatToStr Convert a floating point value to a string
22810: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
22811: Procedure System Flush Flushes buffered text file data to the file
22812: Keyword For Starts a loop that executes a finite number of times
22813: Function SysUtils ForceDirectories Create a new path of directories
22814: Function SysUtils Format Rich formatting of numbers and text into a string
22815: Function SysUtils FormatCurr Rich formatting of a currency value into a string
22816: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
22817: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
22818: Function System Frac The fractional part of a floating point number
22819: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
22820: Procedure System FreeMem Free memory storage used by a variable
22821: Keyword System Function Defines a subroutine that returns a value
22822:
22823: G
22824: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
22825: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
22826: Function System GetLastError Gives the error code of the last failing Windows API call
22827: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
22828: Function System GetMem Get a specified number of storage bytes
22829: Keyword Goto Forces a jump to a label, regardless of nesting
22830:
22831: H
22832: Compiler Directive $H Treat string types as AnsiString or ShortString
22833: Compiler Directive $Hints Determines whether Delphi shows compilation hints
22834: Procedure System Halt Terminates the program with an optional dialog
22835: Function System Hi Returns the hi-order byte of a (2 byte) Integer
22836: Function System High Returns the highest value of a type or variable
22837:
22838: I
22839: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
22840: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
```

```
22841: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
22842: Compiler Directive $IfOpt Tests for the state of a Compiler directive
22843: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
22844: Compiler Directive $IOChecks When on, an IO operation error throws an exception
22845: Keyword If Starts a conditional expression to determine what to do next
22846: Keyword Implementation Starts the implementation (code) section of a Unit
22847: Keyword In Used to test if a value is a member of a set
22848: Procedure System Inc Increment an ordinal variable
22849: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
22850: Procedure System Include Include a value in a set variable
22851: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
22852: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
22853: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
22854: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
22855: Function DateUtils IncYear Increments a TDateTime variable by a number of years
22856: Directive Index Principally defines indexed class data properties
22857: Constant Math Infinity Floating point value of infinite size
22858: Keyword Inherited Used to call the parent class constructor or destructor method
22859: Variable System Input Defines the standard input text file
22860: Function Dialogs InputBox Display a dialog that asks for user text input, with default
22861: Function Dialogs InputQuery Display a dialog that asks for user text input
22862: Procedure System Insert Insert a string into another string
22863: Function System Int The integer part of a floating point number as a float
22864: Type System Int64 A 64 bit sized integer - the largest in Delphi
22865: Type System Integer The basic Integer type
22866: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
22867: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
22868: Function SysUtils IntToStr Convert an integer into a string
22869: Function System IOResult Holds the return code of the last I/O operation
22870: Keyword Is Tests whether an object is a certain class or ascendant
22871: Function Math IsInfinite Checks whether a floating point number is infinite
22872: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
22873: Function System IsMultiThread Returns true if the code is running multiple threads
22874: Function Math IsNaN Checks to see if a floating point number holds a real number
22875:
22876: L
22877: Compiler Directive $L Determines what application debug information is built
22878: Compiler Directive $LocalSymbols Determines what application debug information is built
22879: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
22880: Function SysUtils LastDelimiter Find the last position of selected characters in a string
22881: Function System Length Return the number of elements in an array or string
22882: Function System Ln Gives the natural logarithm of a number
22883: Function System Lo Returns the low-order byte of a (2 byte) Integer
22884: Function Math Log10 Gives the log to base 10 of a number
22885: Variable SysUtils LongDateFormat Long version of the date to string format
22886: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
22887: Type System LongInt An Integer whose size is guaranteed to be 32 bits
22888: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
22889: Variable SysUtils LongTimeFormat Long version of the time to string format
22890: Type System LongWord A 32 bit unsigned integer
22891: Function System Low Returns the lowest value of a type or variable
22892: Function SysUtils LowerCase Change upper case characters in a string to lower case
22893:
22894: M
22895: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
22896: Function Math Max Gives the maximum of two integer values
22897: Constant System MaxInt The maximum value an Integer can have
22898: Constant System MaxLongInt The maximum value an LongInt can have
22899: Function Math Mean Gives the average for a set of numbers
22900: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
22901: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
22902: Function Math Min Gives the minimum of two integer values
22903: Constant SysUtils MinsPerDay Gives the number of minutes in a day
22904: Procedure System MkDir Make a directory
22905: Keyword Mod Performs integer division, returning the remainder
22906: Constant SysUtils MonthDays Gives the number of days in a month
22907: Function DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
22908: Procedure System Move Copy bytes of data from a source to a destination
22909:
22910: N
22911: Constant Math NaN Not a real number
22912: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
22913: Procedure System New Create a new pointer type variable
22914: Constant System Nil A pointer value that is defined as undetermined
22915: Keyword Not Boolean Not or bitwise not of one arguments
22916: Function SysUtils Now Gives the current date and time
22917: Variable Variants Null A variable that has no value
22918:
22919: O
22920: Compiler Directive $O Determines whether Delphi optimises code when compiling
22921: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
22922: Compiler Directive $OverFlowChecks Determines whether Delphi checks integer and enum bounds
22923: Keyword System Object Allows a subroutine data type to refer to an object method
22924: Function System Odd Tests whether an integer has an odd value
22925: Keyword Of Linking keyword used in many places
22926: Keyword On Defines exception handling in a Try Except clause
22927: Keyword Or Boolean or or bitwise or of two arguments
22928: Function System Ord Provides the Ordinal value of an integer, character or enum
22929: Directive Out Identifies a routine parameter for output only
```

```
22930: Variable System Output Defines the standard output text file
22931: Directive Overload Allows 2 or more routines to have the same name
22932: Directive Override Defines a method that replaces a virtual parent class method
22933:
22934: P
22935: Keyword Packed Compacts complex data types into minimal storage
22936: Type System PAnsiChar A pointer to an AnsiChar value
22937: Type System PAnsiString Pointer to an AnsiString value
22938: Function System ParamCount Gives the number of parameters passed to the current program
22939: Function System ParamStr Returns one of the parameters used to run the current program
22940: Type System PChar A pointer to an Char value
22941: Type System PCurrency Pointer to a Currency value
22942: Type System PDateTime Pointer to a TDateTime value
22943: Type System PExtended Pointer to a Extended floating point value
22944: Function System Pi The mathematical constant
22945: Type System PInt64 Pointer to an Int64 value
22946: Function Classes Point Generates a TPoint value from X and Y values
22947: Type System Pointer Defines a general use Pointer to any memory based data
22948: Function Classes PointsEqual Compares two TPoint values for equality
22949: Function System Pos Find the position of one string in another
22950: Function System Pred Decrement an ordinal variable
22951: Function Printers Printer Returns a reference to the global Printer object
22952: Directive Private Starts the section of private data and methods in a class
22953: Keyword System Procedure Defines a subroutine that does not return a value
22954: Procedure FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
22955: Keyword System Program Defines the start of an application
22956: Function Dialogs PromptForFileName Shows a dialog allowing the user to select a file
22957: Keyword System Property Defines controlled access to class fields
22958: Directive Protected Starts a section of class private data accesible to sub-classes
22959: Type System PShortString A pointer to an ShortString value
22960: Type System PString Pointer to a String value
22961: Function Types PtInRect Tests to see if a point lies within a rectangle
22962: Directive Public Starts an externally accessible section of a class
22963: Directive Published Starts a published externally accessible section of a class
22964: Type System PVariant Pointer to a Variant value
22965: Type System PWideChar Pointer to a WideChar
22966: Type System PWideString Pointer to a WideString value
22967:
22968: Q
22969: Compiler Directive $Q Determines whether Delphi checks integer and enum bounds
22970:
22971: R
22972: Compiler Directive $R Determines whether Delphi checks array bounds
22973: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
22974: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
22975: Compiler Directive $Resource Defines a resource file to be included in the application linking
22976: Function Math RadToDeg Converts a radian value to degrees
22977: Keyword Raise Raise an exception
22978: Function System Random Generate a random floating point or integer number
22979: Procedure System Randomize Reposition the Random number generator next value
22980: Function Math RandomRange Generate a random integer number within a supplied range
22981: Variable System RandSeed Reposition the Random number generator next value
22982: Procedure System Read Read data from a binary or text file
22983: Procedure System ReadLn Read a complete line of data from a text file
22984: Type System Real A floating point type supporting about 15 digits of precision
22985: Type System Real48 The floating point type with the highest capacity and precision
22986: Procedure System ReallocMem Reallocate an existing block of storage
22987: Function DateUtils RecodeDate Change only the date part of a TDateTime variable
22988: Function DateUtils RecodeTime Change only the time part of a TDateTime variable
22989: Keyword Record A structured data type - holding fields of data
22990: Function Classes Rect Create a TRect value from 2 points or 4 coordinates
22991: Function SysUtils RemoveDir Remove a directory
22992: Procedure System Rename Rename a file
22993: Function SysUtils RenameFile Rename a file or directory
22994: Keyword Repeat Repeat statements until a ternmination condition is met
22995: Procedure SysUtils ReplaceDate Change only the date part of a TDateTime variable
22996: Procedure SysUtils ReplaceTime Change only the time part of a TDateTime variable
22997: Procedure System Reset Open a text file for reading, or binary file for read/write
22998: Variable System Result A variable used to hold the return value from a function
22999: Procedure System ReWrite Open a text or binary file for write access
23000: Procedure System RmDir Remove a directory
23001: Function System Round Rounds a floating point number to an integer
23002: Procedure System RunError Terminates the program with an error dialog
23003:
23004: S
23005: Constant SysUtils SecsPerDay Gives the number of seconds in a day
23006: Procedure System Seek Move the pointer in a binary file to a new record position
23007: Function System SeekEof Skip to the end of the current line or file
23008: Function System SeekEoln Skip to the end of the current line or file
23009: Function FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
23010: Variable System Self Hidden parameter to a method - refers to the containing object
23011: Keyword Set Defines a set of up to 255 distinct values
23012: Function SysUtils SetCurrentDir Change the current directory
23013: Procedure System SetLength Changes the size of a string, or the size(s) of an array
23014: Procedure System SetString Copies characters from a buffer into a string
23015: Keyword Shl Shift an integer value left by a number of bits
23016: Variable SysUtils ShortDateFormat Compact version of the date to string format
23017: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
23018: Type System ShortInt An integer type supporting values -128 to 127
```

```
23019: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
23020: Type System ShortString Defines a string of up to 255 characters
23021: Variable SysUtils ShortTimeFormat Short version of the time to string format
23022: Procedure Dialogs ShowMessage Display a string in a simple dialog with an OK button
23023: Procedure Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
23024: Procedure Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
23025: Keyword Shr Shift an integer value right by a number of bits
23026: Function System Sin The Sine of a number
23027: Type System Single The smallest capacity and precision floating point type
23028: Function System SizeOf Gives the storage byte size of a type or variable
23029: Function System Slice Creates a slice of an array as an Open Array parameter
23030: Type System SmallInt An Integer type supporting values from -32768 to 32767
23031: Function System Sqr Gives the square of a number
23032: Function System Sqrt Gives the square root of a number
23033: Procedure System Str Converts an integer or floating point number to a string
23034: Type System String A data type that holds a string of characters
23035: Function System StringOfChar Creates a string with one character repeated many times
23036: Function SysUtils StringReplace Replace one or more substrings found within a string
23037: Function System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
23038: Function SysUtils StrScan Searches for a specific character in a constant string
23039: Function SysUtils StrToCurr Convert a number string into a currency value
23040: Function SysUtils StrToDate Converts a date string into a TDateTime value
23041: Function SysUtils StrToDateTime Converts a date+time string into a TDateTime value
23042: Function SysUtils StrToFloat Convert a number string into a floating point value
23043: Function SysUtils StrToInt Convert an integer string into an Integer value
23044: Function SysUtils StrToInt64 Convert an integer string into an Int64 value
23045: Function SysUtils StrToInt64Def Convert a string into an Int64 value with default
23046: Function SysUtils StrToIntDef Convert a string into an Integer value with default
23047: Function SysUtils StrToTime Converts a time string into a TDateTime value
23048: Function StrUtils StuffString Replaces a part of one string with another
23049: Function System Succ Increment an ordinal variable
23050: Function Math Sum Return the sum of an array of floating point values
23051:
23052: T
23053: Function Math Tan The Tangent of a number
23054: Type Classes TBits An object that can hold an infinite number of Boolean values
23055: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
23056: Type ConvUtils TConvType Defines a measurement type as used by Convert
23057: Type System TDateTime Data type holding a date and time value
23058: Type System Text Defines a file as a text file
23059: Type System TextFile Declares a file type for storing lines of text
23060: Type SysUtils TFloatFormat Formats for use in floating point number display functions
23061: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
23062: Keyword Then Part of an if statement - starts the true clause
23063: Variable SysUtils ThousandSeparator The character used to display the thousands separator
23064: Keyword ThreadVar Defines variables that are given separate instances per thread
23065: Function SysUtils Time Gives the current time
23066: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
23067: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
23068: Variable SysUtils TimeSeparator The character used to separate display time fields
23069: Function SysUtils TimeToStr Converts a TDateTime time value to a string
23070: Type Classes TList General purpose container of a list of objects
23071: Keyword To Prefixes an incremental for loop target value
23072: Type System TObject The base class type that is ancestor to all other classes
23073: Function DateUtils Tomorrow Gives the date tomorrow
23074: Type Dialogs TOpenDialog Displays a file selection dialog
23075: Type Types TPoint Holds X and Y integer values
23076: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
23077: Type Types TRect Holds rectangle coordinate values
23078: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
23079: Function SysUtils Trim Removes leading and trailing blanks from a string
23080: Function SysUtils TrimLeft Removes leading blanks from a string
23081: Function SysUtils TrimRight Removes trailing blanks from a string
23082: Function System Trunc The integer part of a floating point number
23083: Procedure System Truncate Truncates a file size - removes all data after the current position
23084: Keyword Try Starts code that has error trapping
23085: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
23086: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
23087: Type Classes TStringList Holds a variable length list of strings
23088: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
23089: Type System TThreadFunc Defines the function to be called by BeginThread
23090: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
23091: Keyword Type Defines a new category of variable or process
23092:
23093: U
23094: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
23095: Keyword Unit Defines the start of a unit file - a Delphi module
23096: Keyword Until Ends a Repeat control loop
23097: Function System UpCase Convert a Char value to upper case
23098: Function SysUtils UpperCase Change lower case characters in a string to upper case
23099: Keyword Uses Declares a list of Units to be imported
23100:
23101: V
23102: Procedure System Val Converts number strings to integer and floating point values
23103: Keyword Var Starts the definition of a section of data variables
23104: Type System Variant A variable type that can hold changing data types
23105: Function Variants VarType Gives the current type of a Variant variable
23106: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
23107: Directive Virtual Allows a class method to be overriden in derived classes
```

```
23108:
23109: W
23110: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
23111: Keyword While Repeat statements whilst a continuation condition is met
23112: Type System WideChar Variable type holding a single International character
23113: Function System WideCharToString Copies a null terminated WideChar string to a normal string
23114: Type System WideString A data type that holds a string of WideChars
23115: Keyword With A means of simplifying references to structured variables
23116: Type System Word An integer type supporting values 0 to 65535
23117: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
23118: Procedure System Write Write data to a binary or text file
23119: Procedure System WriteLn Write a complete line of data to a text file
23120:
23121: X
23122: Compiler Directive $X Controls some Pascal extension handling
23123: Keyword Xor Boolean Xor or bitwise Xor of two arguments
23124:
23125: Y
23126: Compiler Directive $Y Determines whether application symbol information is built
23127: Function DateUtils Yesterday Gives the date yesterday
23128:
23129: Z
23130: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
23131:
23132:
23133: mapX:
23134:  if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
23135:     writeln('cologne map found');
23136:
23137:     GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
23138:
23139:
23140: SHA1:  maXbox3.exe FF1FBC4C54343B301A55089CC01630434175632E
23141: CRC32: maXbox3.exe B5FCA71
23142: Ref:
23143:     1. writeln(SHA1(Exepath+'\maxbox3.exe'))
23144:     2. shdig: TSHA1Digest;
23145:        shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
23146:                for i:= 0 to 19 do write(BytetoHex(shdig[i]));
23147:
23148:     3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
23149:
23150:   ---- bigbitbox code_cleared_checked----
```