```
 1: ********************************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9.101
 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: 23336960 V3.9.9.101 Nov 2014 To EKON/BASTA/JAX/IBZ/SWS/EU/DT/PASCON
10: ****************Now the Funclist****************************************
11: Funclist Function : 13685 //10766 //10165 (7648)
12: ****************Now the Proclist****************************************
13: Proclist Procedure Size is: 8396  //6792 //6401 4752
14: ****************Now Constructors****************************************
15: Constructlist Constructor Size is: 1372 //995 //
16: def head:max: maXbox7: 24.10.2014 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: 23452! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 23850
22: ASize of EXE: 23336960 (22041600) (16586240) (13511680)
23: SHA1 Hash of maXbox 3.9.9.101: 786FA3035226AC3F62F6F6B9E2E2B94C1D659297
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 CheckCom(AComNumber: Integer): Integer;');
351: Function CheckLPT1: string;');
352: function ChrA(const a: byte): char;
353: Function ClassIDToProgID(const ClassID: TGUID): string;
354: Function ClassNameIs(const Name: string): Boolean
355: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
356: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
```

```
357: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
358: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;
359: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;
360: Function ClearBit5( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
361: Function ClearBit6( const Value : Int64; const Bit : TBitRange) : Int64;
362: function CLIENTTOSCREEN(POINT:TPOINT):TPOINT
363: Function Clipboard : TClipboard
364: Function ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
365: Function ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
366: Function ClipLine( var X1, Y1, X2, Y2 : Integer; const ClipRect : TRect) : Boolean;
367: Function ClipLineToRect( var P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean
368: Function Clone( out stm : IStream) : HResult
369: Function CloneConnection : TSQLConnection
370: Function CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
371: function CLOSEQUERY:BOOLEAN
372: Function CloseVolume( var Volume : THandle) : Boolean
373: Function CloseHandle(Handle: Integer): Integer; stdcall;
374: Function CPlApplet( hwndCPl : THandle; uMsg : DWORD; lParam1, lParam2 : Longint) : Longint
375: Function CmdLine: PChar;
376: function CmdShow: Integer;
377: // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
378: Function CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TPos): string;
379: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
380: Function Color32( WinColor : TColor) : TColor32;
381: Function Color321( const Index : Byte; const Palette : TPalette32) : TColor32;
382: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale :BOOLean) : TColor
383: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
384: Function ColorToHTML( const Color : TColor) : String
385: function ColorToIdent(Color: Longint; var Ident: string): Boolean)
386: Function ColorToRGB(color: TColor): Longint
387: function ColorToString(Color: TColor): string)
388: Function ColorToWebColorName( Color : TColor) : string
389: Function ColorToWebColorStr( Color : TColor) : string
390: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn
391: Function Combination(npr, ncr: integer): extended;
392: Function CombinationInt(npr, ncr: integer): Int64;
393: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
394: Function CommaAdd( const AStr1, AStr2 : String) : string
395: Function CommercialRound( const X : Extended) : Int64
396: Function Commit( grfCommitFlags : Longint) : HResult
397: Function Compare( const NameExt : string) : Boolean
398: function CompareDate(const A, B: TDateTime): TValueRelationship;
399: Function CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Integer
400: Function CompareFiles(const FN1,FN2 :string; Breathe:TNotifyEvent;BreathingSender:TObject): boolean
401: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean
402: Function CompareStr( S1, S2 : string) : Integer
403: function CompareStr(const S1: string; const S2: string): Integer)
404: function CompareString(const S1: string; const S2: string): Integer)
405: Function CompareText( S1, S2 : string) : Integer
406: function CompareText(const S1: string; const S2: string): Integer)
407: Function CompareTextLike(cWildStr,cStr:string;const cWildChar:char;lCaseSensitive:boolean): boolean
408: function CompareTime(const A, B: TDateTime): TValueRelationship;
409: function CompareValueE(const A, B: Extended; Epsilon: Extended = 0): TValueRelationship; overload;
410: function CompareValueD(const A, B: Double; Epsilon: Double = 0): TValueRelationship; overload;
411: function CompareValueS(const A, B: Single; Epsilon: Single = 0): TValueRelationship; overload;
412: function CompareValueI(const A, B: Integer): TValueRelationship; overload;
413: function CompareValueI64(const A, B: Int64): TValueRelationship; overload;
414: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
415: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
416: Function ComponentTypeToString( const ComponentType : DWORD) : string
417: Function ComposeDateTime(Date,Time : TDateTime) : TDateTime;');
418: Function ComponentToStringProc(Component: TComponent): string;
419: Function StringToComponentProc(Value: string): TComponent;
420: Function CompToCurrency( Value : Comp) : Currency
421: Function CompToDouble( Value : Comp) : Double
422: function ComputeFileCRC32(const FileName : String) : Integer;
423: function ComputeSHA256(astr: string; amode: char): string)  //mode F:File, S:String
424: function ComputeSHA512(astr: string; amode: char): string)  //mode F:File, S:String
425: function ComPortSelect: Integer;  // Search for the first available port
426: Function Concat(s: string): string
427: Function ConnectAndGetAll : string
428: Function Connected : Boolean
429: function constrain(x, a, b: integer): integer;
430: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
431: Function ConstraintsDisabled : Boolean
432: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
433: Function ContainsState( oState : TniRegularExpressionState) : boolean
434: Function ContainsStr( const AText, ASubText : string) : Boolean
435: Function ContainsText( const AText, ASubText : string) : Boolean
436: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean
437: Function Content : string
438: Function ContentFromStream( Stream : TStream) : string
439: Function ContentFromString( const S : string) : string
440: Function CONTROLSDISABLED : BOOLEAN
441: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
442: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
443: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
444: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
445: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
```

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

```
524: Function CreateTwoColorsBrushPattern( Color1, Color2 : TColor) : TBitmap
525: Function CreateValueBuffer( Length : Integer) : TValueBuffer
526: Function CreateHexDump( AOwner : TWinControl) : THexDump
527: Function Csc( const X : Extended) : Extended
528: Function CscH( const X : Extended) : Extended
529: function currencyDecimals: Byte
530: function currencyFormat: Byte
531: function currencyString: String
532: Function CurrentProcessId : TIdPID
533: Function CurrentReadBuffer : string
534: Function CurrentThreadId : TIdPID
535: Function CurrentYear : Word
536: Function CurrToBCD(const Curr: Currency; var BCD: TBcd; Precision: Integer; Decimals: Integer): Boolean
537: Function CurrToStr( Value : Currency) : string;
538: Function CurrToStrF( Value : Currency; FormatSettings : TFormatSettings; Digits: Integer) : string;
539: Function CurrToStrFS(Value:Currency;Format:TFloatFormat;Digits:Int;const
     FormatSettings:TFormatSettings):string;
540: function CursorToIdent(cursor: Longint; var Ident: string): Boolean;
541: function CursorToString(cursor: TCursor): string;
542: Function CustomSort( SortProc : TLVCompare; lParam : Longint) : Boolean
543: Function CustomSort( SortProc : TTVCompare; Data : Longint; ARecurse : Boolean) : Boolean
544: Function CycleToDeg( const Cycles : Extended) : Extended
545: Function CycleToGrad( const Cycles : Extended) : Extended
546: Function CycleToRad( const Cycles : Extended) : Extended
547: Function D2H( N : Longint; A : Byte) : string
548: Function DarkColor( const Color : TColor; const Pct : Single) : TColor
549: Function DarkColorChannel( const Channel : Byte; const Pct : Single) : Byte
550: Function DataLinkDir : string
551: Function DataRequest( Data : OleVariant) : OleVariant
552: Function DataRequest( Input : OleVariant) : OleVariant
553: Function DataToRawColumn( ACol : Integer) : Integer
554: Function Date : TDateTime
555: function Date: TDateTime;
556: Function DateIsNull( const pdtValue : TDateTime; const pdtKind : TdtKind) : Boolean
557: Function DateOf( const AValue : TDateTime) : TDateTime
558: function DateSeparator: char;
559: Function DateTimeGMTToHttpStr( const GMTValue : TDateTime) : String
560: Function DateTimeToFileDate( DateTime : TDateTime) : Integer
561: function DateTimeToFileDate(DateTime: TDateTime): Integer;
562: Function DateTimeToGmtOffSetStr( ADateTime : TDateTime; SubGMT : Boolean) : string
563: Function DateTimeToInternetStr( const Value : TDateTime; const AIsGMT : Boolean) : String
564: Function DateTimeToJulianDate( const AValue : TDateTime) : Double
565: Function DateTimeToModifiedJulianDate( const AValue : TDateTime) : Double
566: Function DateTimeToStr( DateTime : TDateTime) : string;
567: Function DateTimeToStr2( DateTime : TDateTime; FormatSettings : TFormatSettings) : string;
568: function DateTimeToTimeStamp(DateTime: TDateTime): TTimeStamp
569: Function DateTimeToUnix( const AValue : TDateTime) : Int64
570: function DateTimeToUnix(D: TDateTime): Int64;
571: Function DateToStr( DateTime : TDateTime) : string;
572: function DateToStr(const DateTime: TDateTime): string;
573: function DateToStr(D: TDateTime): string;
574: Function DateToStr2( DateTime : TDateTime; FormatSettings : TFormatSettings) : string;
575: Function DayOf( const AValue : TDateTime) : Word
576: Function DayOfTheMonth( const AValue : TDateTime) : Word
577: function DayOfTheMonth(const AValue: TDateTime): Word;
578: Function DayOfTheWeek( const AValue : TDateTime) : Word
579: Function DayOfTheYear( const AValue : TDateTime) : Word
580: function DayOfTheYear(const AValue: TDateTime): Word;
581: Function DayOfWeek( DateTime : TDateTime) : Word
582: function DayOfWeek(const DateTime: TDateTime): Word;
583: Function DayOfWeekStr( DateTime : TDateTime) : string
584: Function DaysBetween( const ANow, AThen : TDateTime) : Integer
585: Function DaysInAMonth( const AYear, AMonth : Word) : Word
586: Function DaysInAYear( const AYear : Word) : Word
587: Function DaysInMonth( const AValue : TDateTime) : Word
588: Function DaysInYear( const AValue : TDateTime) : Word
589: Function DaySpan( const ANow, AThen : TDateTime) : Double
590: Function DBUseRightToLeftAlignment( AControl : TControl; AField : TField) : Boolean
591: function DecimalSeparator: char;
592: Function DecLimit( var B : Byte; const Limit : Byte; const Decr : Byte) : Byte;
593: Function DecLimit1( var B : Shortint; const Limit : Shortint; const Decr : Shortint) : Shortint;
594: Function DecLimit2( var B : Smallint; const Limit : Smallint; const Decr : Smallint) : Smallint;
595: Function DecLimit3( var B : Word; const Limit : Word; const Decr : Word) : Word;
596: Function DecLimit4( var B : Integer; const Limit : Integer; const Decr : Integer) : Integer;
597: Function DecLimit5( var B : Cardinal; const Limit : Cardinal; const Decr : Cardinal) : Cardinal;
598: Function DecLimit6( var B : Int64; const Limit : Int64; const Decr : Int64) : Int64;
599: Function DecLimitClamp( var B : Byte; const Limit : Byte; const Decr : Byte) : Byte;
600: Function DecLimitClamp1( var B : Shortint; const Limit : Shortint; const Decr : Shortint) : Shortint;
601: Function DecLimitClamp2( var B : Smallint; const Limit : Smallint; const Decr : Smallint) : Smallint;
602: Function DecLimitClamp3( var B : Word; const Limit : Word; const Decr : Word) : Word;
603: Function DecLimitClamp4( var B : Integer; const Limit : Integer; const Decr : Integer) : Integer;
604: Function DecLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Decr : Cardinal) : Cardinal;
605: Function DecLimitClamp6( var B : Int64; const Limit : Int64; const Decr : Int64) : Int64;
606: Function DecodeDateFully( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
607: Function DecodeSoundexInt( AValue : Integer) : string
608: Function DecodeSoundexWord( AValue : Word) : string
609: Function DefaultAlignment : TAlignment
610: Function DefaultCaption : string
611: Function DefaultColor : TColor
```

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

```
701: Function EncodeString( s : string) : string
702: Function DecodeString( s : string) : string
703: Function EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
704: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
705: Function EndIP : String
706: Function EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
707: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
708: Function EndOfAMonth( const AYear, AMonth : Word) : TDateTime
709: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
710: Function EndOfAYear( const AYear : Word) : TDateTime
711: Function EndOfTheDay( const AValue : TDateTime) : TDateTime
712: Function EndOfTheMonth( const AValue : TDateTime) : TDateTime
713: Function EndOfTheWeek( const AValue : TDateTime) : TDateTime
714: Function EndOfTheYear( const AValue : TDateTime) : TDateTime
715: Function EndPeriod( const Period : Cardinal) : Boolean
716: Function EndsStr( const ASubText, AText : string) : Boolean
717: Function EndsText( const ASubText, AText : string) : Boolean
718: Function EnsureMsgIDBrackets( const AMsgID : String) : String
719: Function EnsureRange( const AValue, AMin, AMax : Integer) : Integer;
720: Function EnsureRange1( const AValue, AMin, AMax : Int64) : Int64;
721: Function EnsureRange2( const AValue, AMin, AMax : Double) : Double;
722: Function EOF: boolean
723: Function EOln: boolean
724: Function EqualRect( const R1, R2 : TRect) : Boolean
725: function EqualRect(const R1, R2: TRect): Boolean)
726: Function Equals( Strings : TWideStrings) : Boolean
727: function Equals(Strings: TStrings): Boolean;
728: Function EqualState( oState : TniRegularExpressionState) : boolean
729: Function ErrOutput: Text)
730: function ExceptionParam: String;
731: function ExceptionPos: Cardinal;
732: function ExceptionProc: Cardinal;
733: function ExceptionToString(er: TIFException; Param: String): String;
734: function ExceptionType: TIFException;
735: Function ExcludeTrailingBackslash( S : string) : string
736: function ExcludeTrailingBackslash(const S: string): string)
737: Function ExcludeTrailingPathDelimiter( const APath : string) : string
738: Function ExcludeTrailingPathDelimiter( S : string) : string
739: function ExcludeTrailingPathDelimiter(const S: string): string)
740: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): boolean;
741: Function ExecProc : Integer
742: Function ExecSQL : Integer
743: Function ExecSQL( ExecDirect : Boolean) : Integer
744: Function Execute : _Recordset;
745: Function Execute : Boolean
746: Function Execute : Boolean;
747: Function Execute( const SQL : string; Params : TParams; Cache : Boolean; cursor : phDBICur) : Integer
748: Function Execute( const SQL : WideString; Params : TParams; ResultSet : TPSResult) : Integer
749: Function Execute( ParentWnd : HWND) : Boolean
750: Function Execute1(constCommText:WideString;const CType:TCommType;const
     ExecuteOpts:TExecuteOpts):_Recordset;
751: Function Execute1( const Parameters : OleVariant) : _Recordset;
752: Function Execute1( ParentWnd : HWND) : Boolean;
753: Function Execute2( var RecordsAffected : Integer; const Parameters : OleVariant) : _Recordset;
754: Function ExecuteAction( Action : TBasicAction) : Boolean
755: Function ExecuteDirect( const SQL : WideString) : Integer
756: Function ExecuteFile(const FileName:string;const Params:string;const DefDir:string;ShowCmd:Int):THandle
757: Procedure ExecuteThread2(afunc:TThreadFunction2;thrOK:boolean);AddTypeS('TThreadFunction2','procedure
758: Function CreateThread2(ThreadFunc: TThreadFunction2) : THandle
759: function ExeFileIsRunning(ExeFile: string): boolean;
760: function ExePath: string;
761: function ExePathName: string;
762: Function Exists( AItem : Pointer) : Boolean
763: Function ExitWindows( ExitCode : Cardinal) : Boolean
764: function Exp(x: Extended): Extended;
765: Function ExpandEnvironmentVar( var Value : string) : Boolean
766: Function ExpandFileName( FileName : string) : string
767: function ExpandFileName(const FileName: string): string)
768: Function ExpandUNCFileName( FileName : string) : string
769: function ExpandUNCFileName(const FileName: string): string)
770: Function ExpJ( const X : Float) : Float;
771: Function Exsecans( X : Float) : Float
772: Function Extract( const AByteCount : Integer) : string
773: Function Extract( Item : TClass) : TClass
774: Function Extract( Item : TComponent) : TComponent
775: Function Extract( Item : TObject) : TObject
776: Function ExtractFileDir( FileName : string) : string
777: function ExtractFileDir(const FileName: string): string)
778: Function ExtractFileDrive( FileName : string) : string
779: function ExtractFileDrive(const FileName: string): string)
780: Function ExtractFileExt( FileName : string) : string
781: function ExtractFileExt(const FileName: string): string)
782: Function ExtractFileExtNoDot( const FileName : string) : string
783: Function ExtractFileExtNoDotUpper( const FileName : string) : string
784: Function ExtractFileName( FileName : string) : string
785: function ExtractFileName(const filename: string):string;
786: Function ExtractFilePath( FileName : string) : string
787: function ExtractFilePath(const filename: string):string;
788: Function ExtractRelativePath( BaseName, DestName : string) : string
```

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

```
878: Function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec):Integer)
879: Function FindNext2(var F: TSearchRec): Integer)
880: procedure FindClose2(var F: TSearchRec)
881: Function FINDFIRST : BOOLEAN
882: TJvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
883:   sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms,sfRecent,sfSendTo,
     sfStartMenu, stStartUp, sfTemplates);
884:  FFolder: array [TJvSpecialFolder] of Integer =
885:     (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
886:      CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
887:      CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
888:      CSIDL_STARTUP, CSIDL_TEMPLATES);
889: Function FindFilesDlg(StartIn: string; SpecialFolder: TJvSpecialFolder; UseFolder: Boolean): Boolean);
890: function Findfirst(const filepath: string; attr: integer): integer;
891: function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer)
892: Function FindFirstNotOf( AFind, AText : String) : Integer
893: Function FindFirstOf( AFind, AText : String) : Integer
894: Function FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
895: Function FINDINDEXFORFIELDS( const FIELDS : String) : TINDEXDEF
896: Function FindInstanceOf( AClass : TClass; AExact : Boolean; AStartAt : Integer) : Integer
897: Function FINDITEM( VALUE : INTEGER; KIND : TFINDITEMKIND) : TMENUITEM
898: function FindItemId( Id : Integer) : TCollectionItem
899: Function FindKey( const KeyValues : array of const) : Boolean
900: Function FINDLAST : BOOLEAN
901: Function FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
902: Function FindModuleClass( AClass : TComponentClass) : TComponent
903: Function FindModuleName( const AClass : string) : TComponent
904: Function FINDNEXT : BOOLEAN
905: function FindNext: integer;
906: function FindNext2(var F: TSearchRec): Integer)
907: Function FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible : Boolean) : TTabSheet
908: Function FindNextToSelect : TTreeNode
909: Function FINDPARAM( const VALUE : String) : TPARAM
910: Function FindParam( const Value : WideString) : TParameter
911: Function FINDPRIOR : BOOLEAN
912: Function FindResource( ModuleHandle : HMODULE; ResourceName, ResourceType : PChar) : TResourceHandle
913: Function FindSession( const SessionName : string) : TSession
914: function FindStringResource(Ident: Integer): string)
915: Function FindText( const SearchStr:string;StartPos,Length: Integer; Options: TSearchTypes):Integer
916: Function FindUnusedFileName( const FileName, FileExt, Suffix : AnsiString) : AnsiString
917: function FindVCLWindow(const Pos: TPoint): TWinControl;
918: function FindWindow(C1, C2: PChar): Longint;
919: Function FindInPaths(const fileName,paths: String): String;
920: Function Finger : String
921: Function First : TClass
922: Function First : TComponent
923: Function First : TObject
924: Function FirstDelimiter( const delimiters : string; const Str : String) : integer;
925: Function FirstDelimiter1( const delimiters : WideString; const Str : WideString) : integer;
926: Function FirstInstance( const ATitle : string) : Boolean
927: Function FloatPoint( const X, Y : Float) : TFloatPoint;
928: Function FloatPoint1( const P : TPoint) : TFloatPoint;
929: Function FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) : Boolean
930: Function FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
931: Function FloatRect1( const Rect : TRect) : TFloatRect;
932: Function FloatsEqual( const X, Y : Float) : Boolean
933: Function FloatToBin(const D: Double): string;     //doubletohex -> hextobin! in buffer
934: Function FloatToCurr( Value : Extended) : Currency
935: Function FloatToDateTime( Value : Extended) : TDateTime
936: Function FloatToStr( Value : Extended) : string;
937: Function FloatToStr(e : Extended) : String;
938: Function FloatToStrF( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer) : string;
939: function FloatToStrF(Value: Extended; Format: TFloatFormat;Precision:Integer; Digits: Integer): string)
940: Function FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings
     : TFormatSettings) : string;
941: Function FloatToStrFS( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer;
     FormatSettings : TFormatSettings) : string;
942: function FloatToText(BufferArg: PChar; const Value: Extended; ValueType: TFloatValue;Format: TFloatFormat;
     Precision,Digits: Integer): Integer)
943: Function Floor( const X : Extended) : Integer
944: Function FloorInt( Value : Integer; StepSize : Integer) : Integer
945: Function FloorJ( const X : Extended) : Integer
946: Function Flush( const Count : Cardinal) : Boolean
947: Function Flush(var t: Text): Integer
948: function FmtLoadStr(Ident: Integer; const Args: array of const): string)
949: function FOCUSED:BOOLEAN
950: Function ForceBackslash( const PathName : string) : string
951: Function ForceDirectories( const Dir : string) : Boolean
952: Function ForceDirectories( Dir : string) : Boolean
953: Function ForceDirectories( Name : string) : Boolean
954: Function ForceInBox( const Point : TPoint; const Box : TRect) : TPoint
955: Function ForceInRange( A, Min, Max : Integer) : Integer
956: Function ForceInRangeR( const A, Min, Max : Double) : Double
957: Function ForEach( AProc : TBucketProc; AInfo : Pointer) : Boolean;
958: Function ForEach1( AEvent : TBucketEvent) : Boolean;
959: Function ForegroundTask: Boolean
960: function Format(const Format: string; const Args: array of const): string;
961: Function FormatBcd( const Format : string; Bcd : TBcd) : string
962: FUNCTION FormatBigInt(s: string): STRING;
```

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

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

```
1139: Function GetFromClipboard : Boolean
1140: Function GetFullURI( const AOptionalFileds : TIdURIOptionalFieldsSet) : String
1141: Function GetGBitmap( Value : TBitmap) : TBitmap
1142: Function GetGMTDateByName( const AFileName : TIdFileName) : TDateTime
1143: Function GetGroupState( Level : Integer) : TGroupPosInds
1144: Function GetHandle : HWND
1145: Function GETHELPCONTEXT( VALUE : INTEGER; BYCOMMAND : BOOLEAN) : THELPCONTEXT
1146: Function GetHexArray(ahexdig: THexArray): THexArray;
1147: Function GetHighLightColor( const Color : TColor; Luminance : Integer) : TColor
1148: function GetHINSTANCE: longword;
1149: Function GetHistoryFolder : string
1150: Function GetHitTestInfoAt( X, Y : Integer) : THitTests
1151: function getHMODULE: longword;
1152: Function GetHostByName(const AComputerName: String): String;
1153: Function GetHostName : string
1154: Function getHostIP: string;
1155: Function GetHotSpot : TPoint
1156: Function GetHueBitmap( Value : TBitmap) : TBitmap
1157: Function GetImageBitmap : HBITMAP
1158: Function GETIMAGELIST : TCUSTOMIMAGELIST
1159: Function GetIncome( const aNetto : Currency) : Currency
1160: Function GetIncome( const aNetto : Extended) : Extended
1161: Function GetIncome( const aNetto : Extended): Extended
1162: Function GetIncome(const aNetto : Extended) : Extended
1163: function GetIncome(const aNetto: Currency): Currency
1164: Function GetIncome2( const aNetto : Currency) : Currency
1165: Function GetIncome2( const aNetto : Currency): Currency
1166: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string) : string
1167: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN) : TINDEXDEF
1168: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet) : TIndexDef
1169: Function GetInstRes(Instance:THandle;ResType:TResType;const
      Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1170: Function
      GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1171: Function GetIntelCacheDescription( const D : Byte) : string
1172: Function GetInteractiveUserName : string
1173: Function GetInternetCacheFolder : string
1174: Function GetInternetFormattedFileTimeStamp( const AFilename : String) : String
1175: Function GetIPAddress( const HostName : string) : string
1176: Function GetIP( const HostName : string) : string
1177: Function GetIPHostByName(const AComputerName: String): String;
1178: Function GetIsAdmin: Boolean;
1179: Function GetItem( X, Y : Integer) : LongInt
1180: Function GetItemAt( X, Y : Integer) : TListItem
1181: Function GetItemHeight(Font: TFont): Integer;
1182: Function GetItemPath( Index : Integer) : string
1183: Function GetKeyFieldNames( List : TStrings) : Integer;
1184: Function GetKeyFieldNames1( List : TWideStrings) : Integer;
1185: Function GetKeyState( const VirtualKey : Cardinal) : Boolean
1186: Function GetLastChild : LongInt
1187: Function GetLastChild : TTreeNode
1188: Function GetLastDelimitedToken( const cDelim : char; const cStr : string) : string
1189: function GetLastError: Integer
1190: Function GetLAT_CONV_FACTOR: double;  //for WGS84 power(1 - 1 / 298.257223563, 2);
1191: Function GetLinesCount(sFileName : String): Integer;
1192: Function GetLoader( Ext : string) : TBitmapLoader
1193: Function GetLoadFilter : string
1194: Function GetLocalComputerName : string
1195: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char) : Char
1196: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string) : string
1197: Function GetLocalUserName : string
1198: Function GetLoginUsername : WideString
1199: function getLongDayNames: string)
1200: Function GetLongHint(const hint: string): string
1201: function getLongMonthNames: string)
1202: Function GetMacAddresses( const Machine : string; const Addresses : TStrings) : Integer
1203: Function GetMainAppWndFromPid( PID : DWORD) : HWND
1204: Function GetMapX(C_form,apath: string; const Data: string): boolean; //c_form: [html/json/xml]
1205: //if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
1206: Procedure GetGEOMap(C_form,apath: string; const Data: string);
1207: Function GetMapXGeoReverse(C_form: string; const lat,long: string): string;
1208: //if GetMapXGeoReverse('XML',topPath,'47.0397826','7.62914761277888') then
1209: Function GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
1210: Function GetMaskBitmap : HBITMAP
1211: Function GetMaxAppAddress : Integer
1212: Function GetMciErrorMessage( const MciErrNo : MCIERROR) : string
1213: Function GetMemoryLoad : Byte
1214: Function GetMIMEDefaultFileExt( const MIMEType : string) : TIdFileName
1215: Function GetMIMETypeFromFile( const AFile : string) : string
1216: Function GetMIMETypeFromFile( const AFile : TIdFileName) : string
1217: Function GetMinAppAddress : Integer
1218: Function GetModule : TComponent
1219: Function GetModuleHandle( ModuleName : PChar) : HMODULE
1220: Function GetModuleName( Module : HMODULE) : string
1221: Function GetModulePath( const Module : HMODULE) : string
1222: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;
1223: Function GetMorseID(InChar : Char): Word;');
1224: Function GetMorseString2(InChar : Char): string;');
1225: Function GetMorseLine(dots: boolean): string;');  //whole table!  {1 or dots}
```

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

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

```
1400: Function GetWindowsTempFolder : string
1401: Function GetWindowsUserID : string
1402: Function GetWindowsVersion : TWindowsVersion
1403: Function GetWindowsVersionString : string
1404: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1405: Function GMTToLocalDateTime( S : string) : TDateTime
1406: Function GotoKey : Boolean
1407: Function GradToCycle( const Grads : Extended) : Extended
1408: Function GradToDeg( const Grads : Extended) : Extended
1409: Function GradToDeg( const Value : Extended) : Extended;
1410: Function GradToDeg1( const Value : Double) : Double;
1411: Function GradToDeg2( const Value : Single) : Single;
1412: Function GradToRad( const Grads : Extended) : Extended
1413: Function GradToRad( const Value : Extended) : Extended;
1414: Function GradToRad1( const Value : Double) : Double;
1415: Function GradToRad2( const Value : Single) : Single;
1416: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1417: Function GreenComponent( const Color32 : TColor32) : Integer
1418: function GUIDToString(const GUID: TGUID): string)
1419: Function HandleAllocated : Boolean
1420: function HandleAllocated: Boolean;
1421: Function HandleRequest : Boolean
1422: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1423: Function HarmonicMean( const X : TDynFloatArray) : Float
1424: Function HasAsParent( Value : TTreeNode) : Boolean
1425: Function HASCHILDDEFS : BOOLEAN
1426: Function HasCurValues : Boolean
1427: Function HasExtendCharacter( const s : UTF8String) : Boolean
1428: Function HasFormat( Format : Word) : Boolean
1429: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1430: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1431: Function HashValue(AStream: TStream): LongWord
1432: Function HashValue(AStream: TStream): Word
1433: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1434: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1435: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1436: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1437: Function HashValue16( const ASrc : string) : Word;
1438: Function HashValue16stream( AStream : TStream) : Word;
1439: Function HashValue32( const ASrc : string) : LongWord;
1440: Function HashValue32Stream( AStream : TStream) : LongWord;
1441: Function HasMergeConflicts : Boolean
1442: Function hasMoreTokens : boolean
1443: Function HASPARENT : BOOLEAN
1444: function HasParent: Boolean
1445: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1446: Function HasUTF8BOM( S : TStream) : boolean;
1447: Function HasUTF8BOM1( S : AnsiString) : boolean;
1448: Function Haversine( X : Float) : Float
1449: Function Head( s : string; const subs : string; var tail : string) : string
1450: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1451: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1452: function HELPJUMP(JUMPID:STRING):BOOLEAN
1453: Function HeronianMean( const a, b : Float) : Float
1454: function HexStrToStr(Value: string): string;
1455: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1456: function HexToBin2(HexNum: string): string;
1457: Function HexToDouble( const Hex : string) : Double
1458: function HexToInt(hexnum: string): LongInt;
1459: function HexToStr(Value: string): string;
1460: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1461: function Hi(vdat: word): byte;
1462: function HiByte(W: Word): Byte)
1463: function High: Int64;
1464: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1465: function HINSTANCE: longword;
1466: function HiWord(l: DWORD): Word)
1467: function HMODULE: longword;
1468: Function HourOf( const AValue : TDateTime) : Word
1469: Function HourOfTheDay( const AValue : TDateTime) : Word
1470: Function HourOfTheMonth( const AValue : TDateTime) : Word
1471: Function HourOfTheWeek( const AValue : TDateTime) : Word
1472: Function HourOfTheYear( const AValue : TDateTime) : Word
1473: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1474: Function HourSpan( const ANow, AThen : TDateTime) : Double
1475: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1476: Function HTMLDecode( const AStr : String) : String
1477: Function HTMLEncode( const AStr : String) : String
1478: Function HTMLEscape( const Str : string) : string
1479: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1480: Function HTTPDecode( const AStr : String) : string
1481: Function HTTPEncode( const AStr : String) : string
1482: Function Hypot( const X, Y : Extended) : Extended
1483: Function IBMax( n1, n2 : Integer) : Integer
1484: Function IBMin( n1, n2 : Integer) : Integer
1485: Function IBRandomString( iLength : Integer) : String
1486: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1487: Function IBStripString( st : String; CharsToStrip : String) : String
1488: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
```

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

```
1578: Function IndexOfString( AList : TStringList; Value : Variant) : Integer
1579: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1580: Function IndyGetHostName : string
1581: Function IndyInterlockedDecrement( var I : Integer) : Integer
1582: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1583: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
1584: Function IndyInterlockedIncrement( var I : Integer) : Integer
1585: Function IndyLowerCase( const A1 : string) : string
1586: Function IndyStrToBool( const AString : String) : Boolean
1587: Function IndyUpperCase( const A1 : string) : string
1588: Function InitCommonControl( CC : Integer) : Boolean
1589: Function InitTempPath : string
1590: Function InMainThread : boolean
1591: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1592: Function Input: Text)
1593: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1594: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1595: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1596: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1597: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1598: Function InquireSignal( RtlSigNum : Integer) : TSignalState
1599: Function InRangeR( const A, Min, Max : Double) : Boolean
1600: function Insert( Index : Integer) : TCollectionItem
1601: Function Insert( Index : Integer) : TComboExItem
1602: Function Insert( Index : Integer) : THeaderSection
1603: Function Insert( Index : Integer) : TListItem
1604: Function Insert( Index : Integer) : TStatusPanel
1605: Function Insert( Index : Integer) : TWorkArea
1606: Function Insert( Index : LongInt; const Text : string) : LongInt
1607: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1608: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1609: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1610: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1611: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
1612: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1613: Function Instance : Longint
1614: function InstanceSize: Longint
1615: Function Int(e : Extended) : Extended;
1616: function Int64ToStr(i: Int64): String;
1617: Function IntegerToBcd( const AValue : Integer) : TBcd
1618: Function Intensity( const Color32 : TColor32) : Integer;
1619: Function Intensity( const R, G, B : Single) : Single;
1620: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
      FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1621: Function InterestRate(NPeriods:Integer;const Payment,PresVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1622: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1623: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1624: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1625: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1626: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1627: Function IntMibToStr( const Value : string) : string
1628: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1629: Function IntToBin( Value : cardinal) : string
1630: Function IntToHex( Value : Integer; Digits : Integer) : string;
1631: function IntToHex(a: integer; b: integer): string;
1632: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1633: function IntToHex64(Value: Int64; Digits: Integer): string)
1634: Function IntTo3Str( Value : Longint; separator: string) : string
1635: Function inttobool( aInt : LongInt) : Boolean
1636: function IntToStr(i: Int64): String;
1637: Function IntToStr64(Value: Int64): string)
1638: function IOResult: Integer
1639: Function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1640: Function IsAccel(VK: Word; const Str: string): Boolean
1641: Function IsAddressInNetwork( Address : String) : Boolean
1642: Function IsAdministrator : Boolean
1643: Function IsAlias( const Name : string) : Boolean
1644: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1645: Function IsASCII( const AByte : Byte) : Boolean;
1646: Function IsASCIILDH( const AByte : Byte) : Boolean;
1647: Function IsAssembly(const FileName: string): Boolean;
1648: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1649: Function IsBinary(const AChar : Char) : Boolean
1650: function IsConsole: Boolean)
1651: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1652: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1653: Function IsDelphiDesignMode : boolean
1654: Function IsDelphiRunning : boolean
1655: Function IsDFAState : boolean
1656: Function IsDirectory( const FileName : string) : Boolean
1657: Function IsDomain( const S : String) : Boolean
1658: function IsDragObject(Sender: TObject): Boolean;
1659: Function IsEditing : Boolean
1660: Function ISEMPTY : BOOLEAN
1661: Function IsEqual( Value : TParameters) : Boolean
1662: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1663: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1664: Function IsFirstNode : Boolean
```

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

```
1754: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1755: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1756: Function ItemRect( Index : Integer) : TRect
1757: function ITEMRECT(INDEX:INTEGER):TRECT
1758: Function ItemWidth( Index : Integer) : Integer
1759: Function JavahashCode(val: string): Integer;
1760: Function JosephusG(n,k: integer; var graphout: string): integer;
1761: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1762: Function JustName(PathName : string) : string;  //in path and ext
1763: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1764: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1765: Function KeepAlive : Boolean
1766: Function KeysToShiftState(Keys: Word): TShiftState;
1767: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1768: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1769: Function KeyboardStateToShiftState: TShiftState; overload;
1770: Function Languages : TLanguages
1771: Function Last : TClass
1772: Function Last : TComponent
1773: Function Last : TObject
1774: Function LastDelimiter( Delimiters, S : string) : Integer
1775: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1776: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1777: Function Latitude2WGS84(lat: double): double;
1778: Function LCM(m,n:longint):longint;
1779: Function LCMJ( const X, Y : Cardinal) : Cardinal
1780: Function Ldexp( const X : Extended; const P : Integer) : Extended
1781: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1782: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1783: function Length: Integer;
1784: Procedure LetFileList(FileList: TStringlist; apath: string);
1785: function lengthmp3(mp3path: string):integer;
1786: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1787: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1788: function LinesCount(sfilename:string):extended;
1789: function TextFileLineCount(const FileName: string): integer;
1790: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1791: function LineStart(Buffer, BufPos: PChar): PChar
1792: function LineStart(Buffer, BufPos: PChar): PChar)
1793: function ListSeparator: char;
1794: function Ln(x: Extended): Extended;
1795: Function LnXP1( const X : Extended) : Extended
1796: function Lo(vdat: word): byte;
1797: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1798: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1799: Function LoadFileAsString( const FileName : string) : string
1800: Function LoadFromFile( const FileName : string) : TBitmapLoader
1801: function LoadFile(const FileName: TFileName): string;
1802: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1803: Function LoadPackage(const Name: string): HMODULE
1804: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1805: Function LoadStr( Ident : Integer) : string
1806: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1807: Function LoadWideStr( Ident : Integer) : WideString
1808: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1809: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1810: Function LockServer( fLock : LongBool) : HResult
1811: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1812: Function Log( const X : Extended) : Extended
1813: Function Log10( const X : Extended) : Extended
1814: Function Log2( const X : Extended) : Extended
1815: function LogBase10(X: Float): Float;
1816: Function LogBase2(X: Float): Float;
1817: Function LogBaseN(Base, X: Float): Float;
1818: Function LogN( const Base, X : Extended) : Extended
1819: Function LogOffOS : Boolean
1820: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1821: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1822: Function LongDateFormat: string;
1823: function LongTimeFormat: string;
1824: Function LongWordToFourChar( ACardinal : LongWord) : string
1825: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1826: Function LookupName( const name : string) : TInAddr
1827: Function LookupService( const service : string) : Integer
1828: function Low: Int64;
1829: Function LowerCase( S : string) : string
1830: Function Lowercase(s : AnyString) : AnyString;
1831: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1832: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1833: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1834: function MainInstance: longword
1835: function MainThreadID: longword
1836: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1837: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1838: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1839: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1840: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1841: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
```

```
1842: Function MakeFile(const FileName: string): integer)');
1843: function MakeLong(A, B: Word): Longint)
1844: Function MakeTempFilename( const APath : String) : string
1845: Function MakeValidFileName( const Str : string) : string
1846: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1847: function MakeWord(A, B: Byte): Word)
1848: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1849: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1850: Function MapValues( Mapping : string; Value : string) : string
1851: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1852: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1853: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1854: Function MaskGetFldSeparator( const EditMask : string) : Integer
1855: Function MaskGetMaskBlank( const EditMask : string) : Char
1856: Function MaskGetMaskSave( const EditMask : string) : Boolean
1857: Function MaskIntlLiteralToChar( IChar : Char) : Char
1858: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1859: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1860: Function MaskString( Mask, Value : String) : String
1861: Function Match( const sString : string) : TniRegularExpressionMatchResul
1862: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1863: Function Matches( const Filename : string) : Boolean
1864: Function MatchesMask( const Filename, Mask : string) : Boolean
1865: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1866: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1867: Function Max( AValueOne, AValueTwo : Integer) : Integer
1868: function Max(const x,y: Integer): Integer;
1869: Function Max1( const B1, B2 : Shortint) : Shortint;
1870: Function Max2( const B1, B2 : Smallint) : Smallint;
1871: Function Max3( const B1, B2 : Word) : Word;
1872: function Max3(const x,y,z: Integer): Integer;
1873: Function Max4( const B1, B2 : Integer) : Integer;
1874: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1875: Function Max6( const B1, B2 : Int64) : Int64;
1876: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1877: Function MaxFloat( const X, Y : Float) : Float
1878: Function MaxFloatArray( const B : TDynFloatArray) : Float
1879: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1880: function MaxIntValue(const Data: array of Integer):Integer)
1881: Function MaxJ( const B1, B2 : Byte) : Byte;
1882: function MaxPath: string;
1883: function MaxValue(const Data: array of Double): Double)
1884: Function MaxCalc( const Formula : string) : Extended  //math expression parser
1885: Procedure MaxCalcF( const Formula : string);   //out to console memo2
1886: function MD5(const fileName: string): string;
1887: Function Mean( const Data : array of Double) : Extended
1888: Function Median( const X : TDynFloatArray) : Float
1889: Function Memory : Pointer
1890: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1891: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1892: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1893: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1894: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1895: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1896: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1897: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1898: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1899: Function MibToId( Mib : string) : string
1900: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1901: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1902: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1903: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1904: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1905: Procedure GetMidiOutputs( const List : TStrings)
1906: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
1907: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
1908: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1909: Function MIDINoteToStr( Note : TMIDINote) : string
1910: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1911: Procedure GetMidiOutputs( const List : TStrings)
1912: Procedure MidiOutCheck( Code : MMResult)
1913: Procedure MidiInCheck( Code : MMResult)
1914: Function MilliSecondOf( const AValue : TDateTime) : Word
1915: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1916: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1917: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1918: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1919: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1920: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
1921: Function MilliSecondOfTheYear( const AValue : TDateTime) : Int64
1922: Function MilliSecondsBetween( const ANow, AThen : TDateTime) : Int64
1923: Function MilliSecondSpan( const ANow, AThen : TDateTime) : Double
1924: Function milliToDateTime( MilliSecond : LongInt) : TDateTime');
1925: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
```

```
1926: Function millis: int64;
1927: Function Min( AValueOne, AValueTwo : Integer) : Integer
1928: Function Min1( const B1, B2 : Shortint) : Shortint;
1929: Function Min2( const B1, B2 : Smallint) : Smallint;
1930: Function Min3( const B1, B2 : Word) : Word;
1931: Function Min4( const B1, B2 : Integer) : Integer;
1932: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1933: Function Min6( const B1, B2 : Int64) : Int64;
1934: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1935: Function MinClientRect : TRect;
1936: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1937: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1938: Function MinFloat( const X, Y : Float) : Float
1939: Function MinFloatArray( const B : TDynFloatArray) : Float
1940: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1941: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1942: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1943: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1944: Function MinIntValue( const Data : array of Integer) : Integer
1945: function MinIntValue(const Data: array of Integer):Integer)
1946: Function MinJ( const B1, B2 : Byte) : Byte;
1947: Function MinuteOf( const AValue : TDateTime) : Word
1948: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1949: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1950: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1951: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1952: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1953: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1954: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1955: Function MinValue( const Data : array of Double) : Double
1956: function MinValue(const Data: array of Double): Double)
1957: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1958: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1959: Function ModFloat( const X, Y : Float) : Float
1960: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1961: Function Modify( const Key : string; Value : Integer) : Boolean
1962: Function ModuleCacheID : Cardinal
1963: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1964: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1965: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1966: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1967: Function MonthOf( const AValue : TDateTime) : Word
1968: Function MonthOfTheYear( const AValue : TDateTime) : Word
1969: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1970: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1971: Function MonthStr( DateTime : TDateTime) : string
1972: Function MouseCoord( X, Y : Integer) : TGridCoord
1973: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1974: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1975: Function MoveNext : Boolean
1976: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1977: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1978: Function Name : string
1979: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1980: function NetworkVolume(DriveChar: Char): string
1981: Function NEWBOTTOMLINE : INTEGER
1982: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1983: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1984: Function NEWLINE : TMENUITEM
1985: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1986: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1987: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
      const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1988: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1989: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
      TMenuItem;AENABLED:BOOL): TMENUITEM
1990: Function NEWTOPLINE : INTEGER
1991: Function Next : TIdAuthWhatsNext
1992: Function NextCharIndex( S : String; Index : Integer) : Integer
1993: Function NextRecordSet : TCustomSQLDataSet
1994: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1995: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1996: Function NextToken : Char
1997: Function nextToken : WideString
1998: function NextToken:Char
1999: Function Norm( const Data : array of Double) : Extended
2000: Function NormalizeAngle( const Angle : Extended) : Extended
2001: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
2002: Function NormalizeRect( const Rect : TRect) : TRect
2003: function NormalizeRect(const Rect: TRect): TRect;
2004: Function Now : TDateTime
2005: function Now2: tDateTime
2006: Function NumProcessThreads : integer
2007: Function NumThreadCount : integer
2008: Function NthDayOfWeek( const AValue : TDateTime) : Word
2009: Function NtProductType : TNtProductType
```

```
2010: Function NtProductTypeString : string
2011: function Null: Variant;
2012: Function NullPoint : TPoint
2013: Function NullRect : TRect
2014: Function Null2Blank(aString:String):String;
2015: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2016: Function NumIP : integer
2017: function Odd(x: Longint): boolean;
2018: Function OffsetFromUTC : TDateTime
2019: Function OffsetPoint( const P, Offset : TPoint) : TPoint
2020: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
2021: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
2022: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
2023: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
2024: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
2025: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
2026: function OpenBit:Integer
2027: Function OpenDatabase : TDatabase
2028: Function OpenDatabase( const DatabaseName : string) : TDatabase
2029: Procedure OpenDir(adir: string);
2030: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
2031: Function OpenMap(const Data: string): boolean;
2032: Function OpenMapX(const Data: string): boolean;
2033: Function OpenObject( Value : PChar) : Boolean;
2034: Function OpenObject1( Value : string) : Boolean;
2035: Function OpenSession( const SessionName : string) : TSession
2036: Function OpenVolume( const Drive : Char) : THandle
2037: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
2038: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
2039: Function OrdToBinary( const Value : Byte) : string;
2040: Function OrdToBinary1( const Value : Shortint) : string;
2041: Function OrdToBinary2( const Value : Smallint) : string;
2042: Function OrdToBinary3( const Value : Word) : string;
2043: Function OrdToBinary4( const Value : Integer) : string;
2044: Function OrdToBinary5( const Value : Cardinal) : string;
2045: Function OrdToBinary6( const Value : Int64) : string;
2046: Function OSCheck( RetVal : Boolean) : Boolean
2047: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
2048: Function OSIdentToString( const OSIdent : DWORD) : string
2049: Function Output: Text)
2050: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
2051: Function Owner : TCustomListView
2052: function Owner : TPersistent
2053: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
2054: Function PadL( pStr : String; pLth : integer) : String
2055: Function Padl(s : AnyString;I : longInt) : AnyString;
2056: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
2057: Function PadR( pStr : String; pLth : integer) : String
2058: Function Padr(s : AnyString;I : longInt) : AnyString;
2059: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
2060: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2061: Function Padz(s : AnyString;I : longInt) : AnyString;
2062: Function PaethPredictor( a, b, c : Byte) : Byte
2063: Function PARAMBYNAME( const VALUE : String) : TPARAM
2064: Function ParamByName( const Value : WideString) : TParameter
2065: Function ParamCount: Integer
2066: Function ParamsEncode( const ASrc : string) : string
2067: function ParamStr(Index: Integer): string)
2068: Function ParseDate( const DateStr : string) : TDateTime
2069: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2070: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2071: Function PathAddExtension( const Path, Extension : string) : string
2072: Function PathAddSeparator( const Path : string) : string
2073: Function PathAppend( const Path, Append : string) : string
2074: Function PathBuildRoot( const Drive : Byte) : string
2075: Function PathCanonicalize( const Path : string) : string
2076: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2077: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2078: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2079: Function PathEncode( const ASrc : string) : string
2080: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2081: Function PathExtractFileNameNoExt( const Path : string) : string
2082: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2083: Function PathGetDepth( const Path : string) : Integer
2084: Function PathGetLongName( const Path : string) : string
2085: Function PathGetLongName2( Path : string) : string
2086: Function PathGetShortName( const Path : string) : string
2087: Function PathIsAbsolute( const Path : string) : Boolean
2088: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2089: Function PathIsDiskDevice( const Path : string) : Boolean
2090: Function PathIsUNC( const Path : string) : Boolean
2091: Function PathRemoveExtension( const Path : string) : string
2092: Function PathRemoveSeparator( const Path : string) : string
2093: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2094: Function Peek : Pointer
2095: Function Peek : TObject
2096: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
```

```
2097: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2098: function Permutation(npr, k: integer): extended;
2099: function PermutationInt(npr, k: integer): Int64;
2100: Function PermutationJ( N, R : Cardinal) : Float
2101: Function Pi : Extended;
2102: Function PiE : Extended;
2103: Function PixelsToDialogsX( const Pixels : Word) : Word
2104: Function PixelsToDialogsY( const Pixels : Word) : Word
2105: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2106: Function Point( X, Y : Integer) : TPoint
2107: function Point(X, Y: Integer): TPoint)
2108: Function PointAssign( const X, Y : Integer) : TPoint
2109: Function PointDist( const P1, P2 : TPoint) : Double;
2110: function PointDist(const P1,P2: TFloatPoint): Double;
2111: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2112: function PointDist2(const P1,P2: TPoint): Double;
2113: Function PointEqual( const P1, P2 : TPoint) : Boolean
2114: Function PointIsNull( const P : TPoint) : Boolean
2115: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2116: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2117: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2118: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2119: Function Pop : Pointer
2120: Function Pop : TObject
2121: Function PopnStdDev( const Data : array of Double) : Extended
2122: Function PopnVariance( const Data : array of Double) : Extended
2123: Function PopulationVariance( const X : TDynFloatArray) : Float
2124: function Pos(SubStr, S: AnyString): Longint;
2125: Function PosEqual( const Rect : TRect) : Boolean
2126: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2127: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2128: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2129: Function Post1( AURL : string; const ASource : TStrings) : string;
2130: Function Post2( AURL : string; const ASource : TStream) : string;
2131: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2132: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2133: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2134: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2135: Function Power( const Base, Exponent : Extended) : Extended
2136: Function PowerBig(aval, n:integer): string;
2137: Function PowerIntJ( const X : Float; N : Integer) : Float;
2138: Function PowerJ( const Base, Exponent : Float) : Float;
2139: Function PowerOffOS : Boolean
2140: Function PreformatDateString( Ps : string) : string
2141: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2142: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2143: Function Printer : TPrinter
2144: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2145: Function ProcessResponse : TIdHTTPWhatsNext
2146: Function ProduceContent : string
2147: Function ProduceContentFromStream( Stream : TStream) : string
2148: Function ProduceContentFromString( const S : string) : string
2149: Function ProgIDToClassID(const ProgID: string): TGUID;
2150: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2151: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2152: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2153: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2154: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2155: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2156: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2157: Function Push( AItem : Pointer) : Pointer
2158: Function Push( AObject : TObject) : TObject
2159: Function Put1( AURL : string; const ASource : TStream) : string;
2160: Function Pythagoras( const X, Y : Extended) : Extended
2161: Function queryDLLInterface( var queryList : TStringList) : TStringList
2162: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2163: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2164: Function queryPerformanceCounter2(mse: int64): int64;
2165: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2166: //Function QueryPerformanceFrequency(mse: int64): boolean;
2167: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2168: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2169: Procedure QueryPerformanceCounter1(var aC: Int64);
2170: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2171: Function Quote( const ACommand : String) : SmallInt
2172: Function QuotedStr( S : string) : string
2173: Function RadToCycle( const Radians : Extended) : Extended
2174: Function RadToDeg( const Radians : Extended) : Extended
2175: Function RadToDeg( const Value : Extended) : Extended;
2176: Function RadToDeg1( const Value : Double) : Double;
2177: Function RadToDeg2( const Value : Single) : Single;
2178: Function RadToGrad( const Radians : Extended) : Extended
2179: Function RadToGrad( const Value : Extended) : Extended;
2180: Function RadToGrad1( const Value : Double) : Double;
2181: Function RadToGrad2( const Value : Single) : Single;
```

```
2182: Function RandG( Mean, StdDev : Extended) : Extended
2183: function Random(const ARange: Integer): Integer;
2184: function random2(a: integer): double
2185: function RandomE: Extended;
2186: function RandomF: Extended;
2187: Function RandomFrom( const AValues : array of string) : string;
2188: Function RandomRange( const AFrom, ATo : Integer) : Integer
2189: function randSeed: longint
2190: Function RawToDataColumn( ACol : Integer) : Integer
2191: Function Read : Char
2192: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2193: function Read(Buffer:String;Count:LongInt):LongInt
2194: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2195: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2196: Function ReadCardinal( const AConvert : boolean) : Cardinal
2197: Function ReadChar : Char
2198: Function ReadClient( var Buffer, Count : Integer) : Integer
2199: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2200: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2201: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2202: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
      Bool):Int
2203: Function ReadInteger( const AConvert : boolean) : Integer
2204: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2205: Function ReadLn : string
2206: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2207: function Readln(question: string): string;
2208: Function readm: string;  //read last line in memo2 - console!
2209: Function ReadLnWait( AFailCount : Integer) : string
2210: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2211: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2212: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2213: Function ReadString( const ABytes : Integer) : string
2214: Function ReadString( const Section, Ident, Default : string) : string
2215: Function ReadString( Count : Integer) : string
2216: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2217: Function ReadTimeStampCounter : Int64
2218: Function RebootOS : Boolean
2219: Function Receive( ATimeOut : Integer) : TReplyStatus
2220: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2221: Function ReceiveLength : Integer
2222: Function ReceiveText : string
2223: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2224: Function ReceiveSerialText: string
2225: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2226: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
      AMilliSec:Word):TDateTime
2227: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2228: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2229: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2230: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2231: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2232: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2233: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2234: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2235: Function Reconcile( const Results : OleVariant) : Boolean
2236: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2237: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2238: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2239: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2240: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2241: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2242: Function RectCenter( const R : TRect) : TPoint
2243: Function RectEqual( const R1, R2 : TRect) : Boolean
2244: Function RectHeight( const R : TRect) : Integer
2245: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2246: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2247: Function RectIntersection( const R1, R2 : TRect) : TRect
2248: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2249: Function RectIsEmpty( const R : TRect) : Boolean
2250: Function RectIsNull( const R : TRect) : Boolean
2251: Function RectIsSquare( const R : TRect) : Boolean
2252: Function RectIsValid( const R : TRect) : Boolean
2253: Function RectsAreValid( R : array of TRect) : Boolean
2254: Function RectUnion( const R1, R2 : TRect) : TRect
2255: Function RectWidth( const R : TRect) : Integer
2256: Function RedComponent( const Color32 : TColor32) : Integer
2257: Function Refresh : Boolean
2258: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2259: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2260: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2261: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2262: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2263: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2264: Function ReleaseHandle : HBITMAP
2265: Function ReleaseHandle : HENHMETAFILE
2266: Function ReleaseHandle : HICON
2267: Function ReleasePalette : HPALETTE
2268: Function RemainderFloat( const X, Y : Float) : Float
```

```
2269: Function Remove( AClass : TClass) : Integer
2270: Function Remove( AComponent : TComponent) : Integer
2271: Function Remove( AItem : Integer) : Integer
2272: Function Remove( AItem : Pointer) : Pointer
2273: Function Remove( AItem : TObject) : TObject
2274: Function Remove( AObject : TObject) : Integer
2275: Function RemoveBackslash( const PathName : string) : string
2276: Function RemoveDF( aString : String) : String  //removes thousand separator
2277: Function RemoveDir( Dir : string) : Boolean
2278: function RemoveDir(const Dir: string): Boolean)
2279: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2280: Function RemoveFileExt( const FileName : string) : string
2281: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2282: Function RenameFile( OldName, NewName : string) : Boolean
2283: function RenameFile(const OldName: string; const NewName: string): Boolean)
2284: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2285: Function ReplaceText( const AText, AFromText, AToText : string) : string
2286: Function Replicate(c : char;I : longInt) : String;
2287: Function Request : TWebRequest
2288: Function ResemblesText( const AText, AOther : string) : Boolean
2289: Function Reset : Boolean
2290: function Reset2(mypath: string):string;
2291: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2292: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2293: Function Response : TWebResponse
2294: Function ResumeSupported : Boolean
2295: Function RETHINKHOTKEYS : BOOLEAN
2296: Function RETHINKLINES : BOOLEAN
2297: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2298: Function RetrieveCurrentDir : string
2299: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2300: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2301: Function RetrieveMailBoxSize : integer
2302: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2303: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2304: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2305: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2306: Function ReverseBits( Value : Byte) : Byte;
2307: Function ReverseBits1( Value : Shortint) : Shortint;
2308: Function ReverseBits2( Value : Smallint) : Smallint;
2309: Function ReverseBits3( Value : Word) : Word;
2310: Function ReverseBits4( Value : Cardinal) : Cardinal;
2311: Function ReverseBits4( Value : Integer) : Integer;
2312: Function ReverseBits5( Value : Int64) : Int64;
2313: Function ReverseBytes( Value : Word) : Word;
2314: Function ReverseBytes1( Value : Smallint) : Smallint;
2315: Function ReverseBytes2( Value : Integer) : Integer;
2316: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2317: Function ReverseBytes4( Value : Int64) : Int64;
2318: Function ReverseString( const AText : string) : string
2319: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var
      HostName:String):Bool;
2320: Function Revert : HResult
2321: Function RGB(R,G,B: Byte): TColor;
2322: Function RGB2BGR( const Color : TColor) : TColor
2323: Function RGB2TColor( R, G, B : Byte) : TColor
2324: Function RGBToWebColorName( RGB : Integer) : string
2325: Function RGBToWebColorStr( RGB : Integer) : string
2326: Function RgbToHtml( Value : TColor) : string
2327: Function HtmlToRgb(const Value: string): TColor;
2328: Function RightStr( const AStr : String; Len : Integer) : String
2329: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2330: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2331: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2332: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2333: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2334: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2335: Function Round(e : Extended) : Longint;
2336: Function Round64(e: extended): Int64;
2337: Function RoundAt( const Value : string; Position : SmallInt) : string
2338: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2339: Function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended;');
2340: Function SimpleRoundTo(const AValue: Extended; const ADigit: TRoundToRange): Extended;');
2341: Function RoundFrequency( const Frequency : Integer) : Integer
2342: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2343: Function RoundPoint( const X, Y : Double) : TPoint
2344: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2345: Function RowCount : Integer
2346: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2347: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2348: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2349: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2350: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2351: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2352: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2353: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2354: Function RunByteCode(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2355: Function RunCompiledScript2(Bytecode: AnsiString; out RuntimeErrors: AnsiString): Boolean;');
2356: Function S_AddBackSlash( const ADirName : string) : string
```

```
2357: Function S_AllTrim( const cStr : string) : string
2358: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2359: Function S_Cut( const cStr : string; const iLen : integer) : string
2360: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2361: Function S_DirExists( const ADir : string) : Boolean
2362: Function S_Empty( const cStr : string) : boolean
2363: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2364: Function S_LargeFontsActive : Boolean
2365: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2366: Function S_LTrim( const cStr : string) : string
2367: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2368: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2369: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2370: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2371: Function S_RTrim( const cStr : string) : string
2372: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2373: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2374: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2375: Function S_Space( const iLen : integer) : String
2376: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2377: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2378: Function S_StrCRC32( const Text : string) : LongWORD
2379: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2380: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2381: Function S_StringtoUTF_8( const AString : string) : string
2382: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2383: function S_StrToReal(const cStr: string; var R: Double): Boolean
2384: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2385: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2386: Function S_UTF_8ToString( const AString : string) : string
2387: Function S_WBox( const AText : string) : integer
2388: Function SameDate( const A, B : TDateTime) : Boolean
2389: function SameDate(const A, B: TDateTime): Boolean;
2390: Function SameDateTime( const A, B : TDateTime) : Boolean
2391: function SameDateTime(const A, B: TDateTime): Boolean;
2392: Function SameFileName( S1, S2 : string) : Boolean
2393: Function SameText( S1, S2 : string) : Boolean
2394: function SameText(const S1: string; const S2: string): Boolean)
2395: Function SameTime( const A, B : TDateTime) : Boolean
2396: function SameTime(const A, B: TDateTime): Boolean;
2397: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2398: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2399: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2400: Function SampleVariance( const X : TDynFloatArray) : Float
2401: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2402: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2403: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2404: Function SaveToFile( const AFileName : TFileName) : Boolean
2405: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2406: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2407: Function ScanF(const aformat: String; const args: array of const): string;
2408: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2409: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2410: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2411: function SearchRecattr: integer;
2412: function SearchRecExcludeAttr: integer;
2413: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2414: function SearchRecname: string;
2415: function SearchRecsize: integer;
2416: function SearchRecTime: integer;
2417: Function Sec( const X : Extended) : Extended
2418: Function Secant( const X : Extended) : Extended
2419: Function SecH( const X : Extended) : Extended
2420: Function SecondOf( const AValue : TDateTime) : Word
2421: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2422: Function SecondOfTheHour( const AValue : TDateTime) : Word
2423: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2424: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2425: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2426: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2427: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2428: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2429: Function SectionExists( const Section : string) : Boolean
2430: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2431: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2432: function Seek(Offset:LongInt;Origin:Word):LongInt
2433: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2434: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2435: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2436: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2437: Function SendBuf( var Buf, Count : Integer) : Integer
2438: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2439: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2440: Function SendKey( AppName : string; Key : Char) : Boolean
2441: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2442: Function SendStream( AStream : TStream) : Boolean
```

```
2443: Function SendStreamThenDrop( AStream : TStream) : Boolean
2444: Function SendText( const S : string) : Integer
2445: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2446: Function SendSerialText(Data: String): cardinal
2447: Function Sent : Boolean
2448: Function ServicesFilePath: string
2449: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2450: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2451: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2452: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2453: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2454: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2455: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2456: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2457: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2458: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2459: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2460: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2461: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2462: Function SetCurrentDir( Dir : string) : Boolean
2463: function SetCurrentDir(const Dir: string): Boolean)
2464: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2465: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2466: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2467: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2468: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2469: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2470: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2471: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2472: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2473: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2474: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2475: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2476: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2477: Function SetLocalTime( Value : TDateTime) : boolean
2478: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2479: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2480: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2481: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2482: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2483: Function SetSize( libNewSize : Longint) : HResult
2484: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2485: Function Sgn( const X : Extended) : Integer
2486: function SHA1(const fileName: string): string;
2487: function SHA256(astr: string; amode: char): string)
2488: function SHA512(astr: string; amode: char): string)
2489: Function ShareMemoryManager : Boolean
2490: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2491: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2492: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2493: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2494: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2495: function ShortDateFormat: string;
2496: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2497: function ShortTimeFormat: string;
2498: function SHOWMODAL:INTEGER
2499: Function ShowModalControl(aControl:TControl;BS:TFormBorderStyle;BI:TBorderIcons;WS:TWindowState;aColor:
      TColor; BW: Integer;Title:String;BeforeShowModal:TNotifyEvent): TModalResult');
2500: Function
      ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2501: function ShowWindow(C1: HWND; C2: integer): boolean;
2502: procedure ShowMemory     //in Dialog
2503: function ShowMemory2: string;
2504: Function ShutDownOS : Boolean
2505: Function Signe( const X, Y : Extended) : Extended
2506: Function Sign( const X : Extended) : Integer
2507: Function Sin(e : Extended) : Extended;
2508: Function sinc( const x : Double) : Double
2509: Function SinJ( X : Float) : Float
2510: Function Size( const AFileName : String) : Integer
2511: function SizeOf: Longint;
2512: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2513: function SlashSep(const Path, S: String): String
2514: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2515: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2516: Function SmallPoint(X, Y: Integer): TSmallPoint)
2517: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2518: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2519: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2520: Function SoundexProc( const AText, AOther : string) : Boolean
2521: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2522: Function SoundexWord( const AText : string) : Word
2523: Function SourcePos : Longint
2524: function SourcePos:LongInt
2525: Function Split0( Str : string; const substr : string) : TStringList
2526: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2527: Function SQLRequiresParams( const SQL : WideString) : Boolean
2528: Function Sqr(e : Extended) : Extended;
```

```
2529: Function Sqrt(e : Extended) : Extended;
2530: Function StartIP : String
2531: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2532: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2533: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2534: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2535: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2536: Function StartOfAYear( const AYear : Word) : TDateTime
2537: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2538: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2539: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2540: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2541: Function StartsStr( const ASubText, AText : string) : Boolean
2542: Function StartsText( const ASubText, AText : string) : Boolean
2543: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2544: Function StartsWith( const str : string; const sub : string) : Boolean
2545: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2546: Function StatusString( StatusCode : Integer) : string
2547: Function StdDev( const Data : array of Double) : Extended
2548: Function Stop : Float
2549: Function StopCount( var Counter : TJclCounter) : Float
2550: Function StoreColumns : Boolean
2551: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2552: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2553: Function StrAlloc( Size : Cardinal) : PChar
2554: function StrAlloc(Size: Cardinal): PChar)
2555: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2556: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2557: Function StrBufSize( Str : PChar) : Cardinal
2558: function StrBufSize(const Str: PChar): Cardinal)
2559: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2560: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2561: Function StrCat( Dest : PChar; Source : PChar) : PChar
2562: function StrCat(Dest: PChar; const Source: PChar): PChar)
2563: Function StrCharLength( Str : PChar) : Integer
2564: Function StrComp( Str1, Str2 : PChar) : Integer
2565: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2566: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2567: Function StrCopy(Dest: PChar; const Source: PChar): PChar)
2568: Function Stream_to_AnsiString( Source : TStream) : ansistring
2569: Function Stream_to_Base64( Source : TStream) : ansistring
2570: Function Stream_to_decimalbytes( Source : TStream) : string
2571: Function Stream2WideString( oStream : TStream) : WideString
2572: Function StreamtoAnsiString( Source : TStream) : ansistring
2573: Function StreamToByte( Source : TStream) : string
2574: Function StreamToDecimalbytes( Source : TStream) : string
2575: Function StreamtoOrd( Source : TStream) : string
2576: Function StreamToString( Source : TStream) : string
2577: Function StreamToString2( Source : TStream) : string
2578: Function StreamToString3( Source : TStream) : string
2579: Function StreamToString4( Source : TStream) : string
2580: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2581: Function StrEmpty( const sString : string) : boolean
2582: Function StrEnd( Str : PChar) : PChar
2583: function StrEnd(const Str: PChar): PChar)
2584: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2585: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2586: Function StrGet(var S : String; I : Integer) : Char;
2587: Function StrGet2(S : String; I : Integer) : Char;
2588: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2589: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2590: Function StrHtmlDecode( const AStr : String) : String
2591: Function StrHtmlEncode( const AStr : String) : String
2592: Function StrToBytes(const Value: String): TBytes;
2593: Function StrIComp( Str1, Str2 : PChar) : Integer
2594: Function StringOfChar(c : char;I : longInt) : String;
2595: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2596: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2597: Function StringRefCount(const s: String): integer;
2598: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2599: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2600: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2601: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2602: Function StringToBoolean( const Ps : string) : Boolean
2603: function StringToColor(const S: string): TColor)
2604: function StringToCursor(const S: string): TCursor;
2605: function StringToGUID(const S: string): TGUID)
2606: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2607: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2608: Function StringWidth( S : string) : Integer
2609: Function StrInternetToDateTime( Value : string) : TDateTime
2610: Function StrIsDateTime( const Ps : string) : Boolean
2611: Function StrIsFloatMoney( const Ps : string) : Boolean
2612: Function StrIsInteger( const S : string) : Boolean
2613: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2614: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2615: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2616: Function StrLen( Str : PChar) : Cardinal
2617: function StrLen(const Str: PChar): Cardinal)
```

```
2618: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2619: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2620: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2621: Function StrLower( Str : PChar) : PChar
2622: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2623: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2624: Function StrNew( Str : PChar) : PChar
2625: function StrNew(const Str: PChar): PChar)
2626: Function StrNextChar( Str : PChar) : PChar
2627: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2628: Function StrParse( var sString : string; const sDelimiters : string) : string;
2629: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2630: Function StrPas( Str : PChar) : string
2631: function StrPas(const Str: PChar): string)
2632: Function StrPCopy( Dest : PChar; Source : string) : PChar
2633: function StrPCopy(Dest: PChar; const Source: string): PChar)
2634: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2635: Function StrPos( Str1, Str2 : PChar) : PChar
2636: Function StrScan(const Str: PChar; Chr: Char): PChar)
2637: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2638: Function StrToBcd( const AValue : string) : TBcd
2639: Function StrToBool( S : string) : Boolean
2640: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2641: Function StrToCard( const AStr : String) : Cardinal
2642: Function StrToConv( AText : string; out AType : TConvType) : Double
2643: Function StrToCurr( S : string) : Currency;
2644: function StrToCurr(const S: string): Currency)
2645: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2646: Function StrToDate( S : string) : TDateTime;
2647: function StrToDate(const s: string): TDateTime;
2648: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2649: Function StrToDateTime( S : string) : TDateTime;
2650: function StrToDateTime(const S: string): TDateTime)
2651: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2652: Function StrToDay( const ADay : string) : Byte
2653: Function StrToFloat( S : string) : Extended;
2654: function StrToFloat(s: String): Extended;
2655: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2656: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2657: Function StrToFloat( S : string) : Extended;
2658: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2659: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2660: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2661: Function StrToCurr( S : string) : Currency;
2662: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2663: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2664: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2665: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2666: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2667: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2668: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2669: Function StrToDateTime( S : string) : TDateTime;
2670: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2671: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2672: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2673: Function StrToInt( S : string) : Integer
2674: function StrToInt(s: String): Longint;
2675: Function StrToInt64( S : string) : Int64
2676: function StrToInt64(s: String): int64;
2677: Function StrToInt64Def( S : string; Default : Int64) : Int64
2678: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2679: Function StrToIntDef( S : string; Default : Integer) : Integer
2680: function StrToIntDef(const S: string; Default: Integer): Integer)
2681: function StrToIntDef(s: String; def: Longint): Longint;
2682: Function StrToMonth( const AMonth : string) : Byte
2683: Function StrToTime( S : string) : TDateTime;
2684: function StrToTime(const S: string): TDateTime)
2685: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2686: Function StrToWord( const Value : String) : Word
2687: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2688: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2689: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2690: Function StrUpper( Str : PChar) : PChar
2691: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2692: Function Sum( const Data : array of Double) : Extended
2693: Function SumFloatArray( const B : TDynFloatArray) : Float
2694: Function SumInt( const Data : array of Integer) : Integer
2695: Function SumOfSquares( const Data : array of Double) : Extended
2696: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2697: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2698: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2699: Function Supports( CursorOptions : TCursorOptions) : Boolean
2700: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2701: Function SwapWord(w : word): word)
2702: Function SwapInt(i : integer): integer)
2703: Function SwapLong(L : longint): longint)
2704: Function Swap(i : integer): integer)
2705: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2706: Function SyncTime : Boolean
```

```
2707: Function SysErrorMessage( ErrorCode : Integer) : string
2708: function SysErrorMessage(ErrorCode: Integer): string)
2709: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2710: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2711: Function SysStringLen(const S: WideString): Integer; stdcall;
2712: Function TabRect( Index : Integer) : TRect
2713: Function Tan( const X : Extended) : Extended
2714: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2715: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2716: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2717: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2718: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2719: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2720: Function TenToY( const Y : Float) : Float
2721: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2722: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2723: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2724: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2725: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2726: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2727: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2728: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2729: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2730: Function TestBits( const Value, Mask : Byte) : Boolean;
2731: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2732: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2733: Function TestBits3( const Value, Mask : Word) : Boolean;
2734: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2735: Function TestBits5( const Value, Mask : Integer) : Boolean;
2736: Function TestBits6( const Value, Mask : Int64) : Boolean;
2737: Function TestFDIVInstruction : Boolean
2738: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2739: Function TextExtent( const Text : string) : TSize
2740: function TextHeight(Text: string): Integer;
2741: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2742: Function TextStartsWith( const S, SubS : string) : Boolean
2743: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2744: Function ConvInteger(i : integer):string;
2745: Function IntegerToText(i : integer):string;
2746: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2747: function TextWidth(Text: string): Integer;
2748: Function ThreadCount : integer
2749: function ThousandSeparator: char;
2750: Function Ticks : Cardinal
2751: Function Time : TDateTime
2752: function Time: TDateTime;
2753: function TimeGetTime: int64;
2754: Function TimeOf( const AValue : TDateTime) : TDateTime
2755: function TimeSeparator: char;
2756: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2757: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2758: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2759: Function TimeToStr( DateTime : TDateTime) : string;
2760: function TimeToStr(const DateTime: TDateTime): string;
2761: Function TimeZoneBias : TDateTime
2762: Function ToCommon( const AValue : Double) : Double
2763: function ToCommon(const AValue: Double): Double;
2764: Function Today : TDateTime
2765: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2766: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2767: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2768: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2769: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2770: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2771: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2772: function TokenComponentIdent:String
2773: Function TokenFloat : Extended
2774: function TokenFloat:Extended
2775: Function TokenInt : Longint
2776: function TokenInt:LongInt
2777: Function TokenString : string
2778: function TokenString:String
2779: Function TokenSymbolIs( const S : string) : Boolean
2780: function TokenSymbolIs(S:String):Boolean
2781: Function Tomorrow : TDateTime
2782: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2783: Function ToString : string
2784: Function TotalVariance( const Data : array of Double) : Extended
2785: Function Trace2( AURL : string) : string;
2786: Function TrackMenu( Button : TToolButton) : Boolean
2787: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2788: Function TranslateURI( const URI : string) : string
2789: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
```

```
2790: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2791: Function Trim( S : string) : string;
2792: Function Trim( S : WideString) : WideString;
2793: Function Trim(s : AnyString) : AnyString;
2794: Function TrimAllOf( ATrim, AText : String) : String
2795: Function TrimLeft( S : string) : string;
2796: Function TrimLeft( S : WideString) : WideString;
2797: function TrimLeft(const S: string): string)
2798: Function TrimRight( S : string) : string;
2799: Function TrimRight( S : WideString) : WideString;
2800: function TrimRight(const S: string): string)
2801: function TrueBoolStrs: array of string
2802: Function Trunc(e : Extended) : Longint;
2803: Function Trunc64(e: extended): Int64;
2804: Function TruncPower( const Base, Exponent : Float) : Float
2805: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2806: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2807: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2808: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2809: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2810: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
      AValue:TDateTime):Boolean
2811: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2812: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
      AVal:TDateTime):Bool
2813: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2814: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2815: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2816: Function TryLock : Boolean
2817: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2818: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
2819: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2820: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2821: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2822: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2823: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2824: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2825: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2826: function TryStrToBool(const S: string; out Value: Boolean): Boolean;
2827: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2828: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2829: Function TwoToY( const Y : Float) : Float
2830: Function UCS4StringToWideString( const S : UCS4String) : WideString
2831: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2832: function Unassigned: Variant;
2833: Function UndoLastChange( FollowChange : Boolean) : Boolean
2834: function UniCodeToStr(Value: string): string;
2835: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2836: Function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2837: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2838: Function UnixPathToDosPath( const Path : string) : string
2839: Function UnixToDateTime( const AValue : Int64) : TDateTime
2840: function UnixToDateTime(U: Int64): TDateTime;
2841: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2842: Function UnlockResource( ResData : HGLOBAL) : LongBool
2843: Function UnlockVolume( var Handle : THandle) : Boolean
2844: Function UnMaskString( Mask, Value : String) : String
2845: function UpCase(ch : Char ) : Char;
2846: Function UpCaseFirst( const AStr : string) : string
2847: Function UpCaseFirstWord( const AStr : string) : string
2848: Function UpdateAction( Action : TBasicAction) : Boolean
2849: Function UpdateKind : TUpdateKind
2850: Function UPDATESTATUS : TUPDATESTATUS
2851: Function UpperCase( S : string) : string
2852: Function Uppercase(s : AnyString) : AnyString;
2853: Function URLDecode( ASrc : string) : string
2854: Function URLEncode( const ASrc : string) : string
2855: Function UseRightToLeftAlignment : Boolean
2856: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2857: Function UseRightToLeftReading : Boolean
2858: Function UTF8CharLength( Lead : Char) : Integer
2859: Function UTF8CharSize( Lead : Char) : Integer
2860: Function UTF8Decode( const S : UTF8String) : WideString
2861: Function UTF8Encode( const WS : WideString) : UTF8String
2862: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2863: Function Utf8ToAnsi( const S : UTF8String) : string
2864: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2865: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2866: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2867: Function ValidParentForm(control: TControl): TForm
2868: Function Value : Variant
2869: Function ValueExists( const Section, Ident : string) : Boolean
2870: Function ValueOf( const Key : string) : Integer
2871: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2872: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2873: Function VarArrayFromStrings( Strings : TStrings) : Variant
2874: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
```

```
2875: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2876: Function VarFMTBcd : TVarType
2877: Function VarFMTBcdCreate1 : Variant;
2878: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2879: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2880: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2881: Function Variance( const Data : array of Double) : Extended
2882: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2883: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2884: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2885: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2886: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2887: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2888: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2889: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2890: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2891: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2892: Function VariantNeg( const V1 : Variant) : Variant
2893: Function VariantNot( const V1 : Variant) : Variant
2894: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2895: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2896: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2897: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2898: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2899: function VarIsEmpty(const V: Variant): Boolean;
2900: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2901: function VarIsNull(const V: Variant): Boolean;
2902: Function VarToBcd( const AValue : Variant) : TBcd
2903: function VarType(const V: Variant): TVarType;
2904: Function VarType( const V : Variant) : TVarType
2905: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2906: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2907: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2908: Function VarIsByRef( const V : Variant) : Boolean
2909: Function VarIsEmpty( const V : Variant) : Boolean
2910: Procedure VarCheckEmpty( const V : Variant)
2911: Function VarIsNull( const V : Variant) : Boolean
2912: Function VarIsClear( const V : Variant) : Boolean
2913: Function VarIsCustom( const V : Variant) : Boolean
2914: Function VarIsOrdinal( const V : Variant) : Boolean
2915: Function VarIsFloat( const V : Variant) : Boolean
2916: Function VarIsNumeric( const V : Variant) : Boolean
2917: Function VarIsStr( const V : Variant) : Boolean
2918: Function VarToStr( const V : Variant) : string
2919: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2920: Function VarToWideStr( const V : Variant) : WideString
2921: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2922: Function VarToDateTime( const V : Variant) : TDateTime
2923: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2924: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2925: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2926: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2927: Function VarSameValue( const A, B : Variant) : Boolean
2928: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2929: Function VarIsEmptyParam( const V : Variant) : Boolean
2930: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2931: Function VarIsError1( const V : Variant) : Boolean;
2932: Function VarAsError( AResult : HRESULT) : Variant
2933: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2934: Function VarIsArray( const A : Variant) : Boolean;
2935: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2936: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2937: Function VarArrayOf( const Values : array of Variant) : Variant
2938: Function VarArrayRef( const A : Variant) : Variant
2939: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2940: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2941: Function VarArrayDimCount( const A : Variant) : Integer
2942: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2943: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2944: Function VarArrayLock( const A : Variant) : ___Pointer
2945: Procedure VarArrayUnlock( const A : Variant)
2946: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2947: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2948: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2949: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2950: Function Unassigned : Variant
2951: Function Null : Variant
2952: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2953: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2954: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2955: function VectorDot(const V1,V2: TFloatPoint): Double;
2956: Function VectorLengthSqr( const V : TFloatPoint) : Double
2957: function VectorLengthSqr(const V: TFloatPoint): Double;
2958: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2959: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2960: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2961: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2962: Function Verify( AUserName : String) : String
2963: Function Versine( X : Float) : Float
```

```
2964: function VersionCheck: boolean;
2965: function VersionCheckAct: string;
2966: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2967: Function VersionLanguageName( const LangId : Word) : string
2968: Function VersionResourceAvailable( const FileName : string) : Boolean
2969: Function Visible : Boolean
2970: function VolumeID(DriveChar: Char): string
2971: Function WaitFor( const AString : string) : string
2972: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2973: Function WaitFor1 : TWaitResult;
2974: Function WaitForData( Timeout : Longint) : Boolean
2975: Function WebColorNameToColor( WebColorName : string) : TColor
2976: Function WebColorStrToColor( WebColor : string) : TColor
2977: Function WebColorToRGB( WebColor : Integer) : Integer
2978: Function wGet(aURL, afile: string): boolean;'
2979: Function wGet2(aURL, afile: string): boolean;'  //without file open
2980: Function wGetX(aURL, afile: string): boolean;
2981: Function wGetX2(aURL, afile: string): boolean;'  //without file open
2982: Function WebGet(aURL, afile: string): boolean;'
2983: Function WebExists: boolean;   //alias to isinternet
2984: Function WeekOf( const AValue : TDateTime) : Word
2985: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2986: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2987: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2988: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2989: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2990: Function WeeksInAYear( const AYear : Word) : Word
2991: Function WeeksInYear( const AValue : TDateTime) : Word
2992: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2993: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2994: Function WideCat( const x, y : WideString) : WideString
2995: Function WideCompareStr( S1, S2 : WideString) : Integer
2996: function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
2997: Function WideCompareText( S1, S2 : WideString) : Integer
2998: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
2999: Function WideCopy( const src : WideString; index, count : Integer) : WideString
3000: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
3001: Function WideEqual( const x, y : WideString) : Boolean
3002: function WideFormat(const Format: WideString; const Args: array of const): WideString)
3003: Function WideGreater( const x, y : WideString) : Boolean
3004: Function WideLength( const src : WideString) : Integer
3005: Function WideLess( const x, y : WideString) : Boolean
3006: Function WideLowerCase( S : WideString) : WideString
3007: function WideLowerCase(const S: WideString): WideString)
3008: Function WidePos( const src, sub : WideString) : Integer
3009: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
3010: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
3011: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
3012: Function WideSameStr( S1, S2 : WideString) : Boolean
3013: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
3014: Function WideSameText( S1, S2 : WideString) : Boolean
3015: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
3016: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
3017: Function WideStringToUCS4String( const S : WideString) : UCS4String
3018: Function WideUpperCase( S : WideString) : WideString
3019: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
3020: function Win32Check(RetVal: boolean): boolean)
3021: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
3022: Function Win32RestoreFile( const FileName : string) : Boolean
3023: Function Win32Type : TIdWin32Type
3024: Function WinColor( const Color32 : TColor32) : TColor
3025: function winexec(FileName: pchar; showCmd: integer): integer;
3026: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3027: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3028: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
3029: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
3030: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
3031: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
3032: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
3033: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
3034: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
3035: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
3036: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
3037: Function WordToStr( const Value : Word) : String
3038: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
3039: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
3040: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
3041: Function WorkArea : Integer
3042: Function WrapText( Line : string; MaxCol : Integer) : string;
3043: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
3044: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3045: function Write(Buffer:String;Count:LongInt):LongInt
3046: Function WriteClient( var Buffer, Count : Integer) : Integer
3047: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
3048: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
3049: Function WriteString( const AString : string) : Boolean
3050: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3051: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
3052: Function wsprintf( Output : PChar; Format : PChar) : Integer
```

```
3053: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3054: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3055: Function XorDecode( const Key, Source : string) : string
3056: Function XorEncode( const Key, Source : string) : string
3057: Function XorString( const Key, Src : ShortString) : ShortString
3058: Function Yield : Bool
3059: Function YearOf( const AValue : TDateTime) : Word
3060: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3061: Function YearSpan( const ANow, AThen : TDateTime) : Double
3062: Function Yesterday : TDateTime
3063: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3064: Function( const Name : string; Proc : TUserFunction)
3065: Function using Special_Scholz from 3.8.5.0
3066: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3067: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3068: Function FloatToTime2Dec(value:Extended):Extended;
3069: Function MinToStd(value:Extended):Extended;
3070: Function MinToStdAsString(value:Extended):String;
3071: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3072: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3073: Function Round2Dec (zahl:Extended):Extended;
3074: Function GetAngle(x,y:Extended):Double;
3075: Function AddAngle(a1,a2:Double):Double;
3076:
3077: ************************************************************************
3078: unit uPSI_StText;
3079: ************************************************************************
3080: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3081: Function TextFileSize( var F : TextFile) : LongInt
3082: Function TextPos( var F : TextFile) : LongInt
3083: Function TextFlush( var F : TextFile) : Boolean
3084:
3085: ************************************************************************
3086: from  JvVCLUtils;
3087: ************************************************************************
3088: { Windows resources (bitmaps and icons) VCL-oriented routines }
3089: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3090: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
      DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3091: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
      Bitmap: TBitmap; TransparentColor: TColor);
3092: function MakeBitmap(ResID: PChar): TBitmap;
3093: function MakeBitmapID(ResID: Word): TBitmap;
3094: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3095: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3096: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3097: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3098:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3099: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3100: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3101: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3102: {$IFDEF WIN32}
3103: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3104:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3105: {$ENDIF}
3106: function MakeIcon(ResID: PChar): TIcon;
3107: function MakeIconID(ResID: Word): TIcon;
3108: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3109: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3110: {$IFDEF WIN32}
3111: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3112: {$ENDIF}
3113: { Service routines }
3114: procedure NotImplemented;
3115: procedure ResourceNotFound(ResID: PChar);
3116: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3117: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3118: function PaletteColor(Color: TColor): Longint;
3119: function WidthOf(R: TRect): Integer;
3120: function HeightOf(R: TRect): Integer;
3121: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3122: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3123: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3124: procedure Delay(MSecs: Longint);
3125: procedure DeleteLine(StrList: TStringList; SearchPattern: String);
3126: procedure CenterControl(Control: TControl);
3127: Function PaletteEntries( Palette : HPALETTE) : Integer
3128: Function WindowClassName( Wnd : HWND) : string
3129: Function ScreenWorkArea : TRect
3130: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3131: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3132: Procedure ActivateWindow( Wnd : HWND)
3133: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3134: Procedure CenterWindow( Wnd : HWND)
3135: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3136: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3137: Function DialogsToPixelsX( Dlgs : Word) : Word
3138: Function DialogsToPixelsY( Dlgs : Word) : Word
3139: Function PixelsToDialogsX( Pixs : Word) : Word
```

```
3140: Function PixelsToDialogsY( Pixs : Word) : Word
3141: {$IFDEF WIN32}
3142: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3143: function MakeVariant(const Values: array of Variant): Variant;
3144: {$ENDIF}
3145: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3146: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3147: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3148: {$IFDEF CBUILDER}
3149: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3150: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3151: {$ELSE}
3152: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3153: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3154: {$ENDIF CBUILDER}
3155: function IsForegroundTask: Boolean;
3156: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3157: function GetAveCharSize(Canvas: TCanvas): TPoint;
3158: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3159: procedure FreeUnusedOle;
3160: procedure Beep;
3161: function GetWindowsVersionJ: string;
3162: function LoadDLL(const LibName: string): THandle;
3163: function RegisterServer(const ModuleName: string): Boolean;
3164: {$IFNDEF WIN32}
3165: function IsLibrary: Boolean;
3166: {$ENDIF}
3167: { Gradient filling routine }
3168: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3169: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
     TFillDirection; Colors: Byte);
3170: { String routines }
3171: function GetEnvVar(const VarName: string): string;
3172: function AnsiUpperFirstChar(const S: string): string;
3173: function StringToPChar(var S: string): PChar;
3174: function StrPAlloc(const S: string): PChar;
3175: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3176: function DropT(const S: string): string;
3177: { Memory routines }
3178: function AllocMemo(Size: Longint): Pointer;
3179: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3180: procedure FreeMemo(var fpBlock: Pointer);
3181: function GetMemoSize(fpBlock: Pointer): Longint;
3182: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3183: {$IFNDEF COMPILER5_UP}
3184: procedure FreeAndNil(var Obj);
3185: {$ENDIF}
3186: // from PNGLoader
3187: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3188: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3189: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3190: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3191: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
     TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3192:   Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3193:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3194:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3195:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3196:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3197:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3198:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3199:  Procedure SetImeName( Name : TImeName)
3200:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3201:  Function Imm32GetContext( hWnd : HWND) : HIMC
3202:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3203:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
3204:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3205:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3206: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3207:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3208:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3209:  Function Imm32IsIME( hKl : longword) : Boolean
3210:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3211:  Procedure DragDone( Drop : Boolean)
3212:
3213:
3214: //**************************************added  from jvjvclutils
3215: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3216: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3217: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3218: function IsPositiveResult(Value: TModalResult): Boolean;
3219: function IsNegativeResult(Value: TModalResult): Boolean;
3220: function IsAbortResult(const Value: TModalResult): Boolean;
3221: function StripAllFromResult(const Value: TModalResult): TModalResult;
3222: // returns either BrightColor or DarkColor depending on the luminance of AColor
3223: // This function gives the same result (AFAIK) as the function used in Windows to
3224: // calculate the desktop icon text color based on the desktop background color
3225: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3226: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
```

```
3227:
3228: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3229:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3230:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3231:   var LinkName: string; Scale: Integer = 100); overload;
3232: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3233:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3234:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3235:   var LinkName: string; Scale: Integer = 100); overload;
3236: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3237:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3238: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3239:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3240:   Scale: Integer = 100): string;
3241: function HTMLPlainText(const Text: string): string;
3242: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3243:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3244: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3245:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3246: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3247: function HTMLPrepareText(const Text: string): string;
3248:
3249: ****************************************** uPSI_JvAppUtils;
3250: Function GetDefaultSection( Component : TComponent) : string
3251: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3252: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3253: Function GetDefaultIniName : string
3254:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3255: Function GetDefaultIniRegKey : string
3256: Function FindForm( FormClass : TFormClass) : TForm
3257: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3258: Function ShowDialog( FormClass : TFormClass) : Boolean
3259:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3260: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3261: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3262: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3263: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3264: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3265: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3266: Function StrToIniStr( const Str : string) : string
3267: Function IniStrToStr( const Str : string) : string
3268: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3269: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3270: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3271: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3272: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3273: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
3274: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3275: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3276: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3277: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3278: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3279: Procedure AppTaskbarIcons( AppOnly : Boolean)
3280: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3281: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3282: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3283: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3284: ****************************************** uPSI_JvDBUtils;
3285: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3286: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3287: Procedure RefreshQuery( Query : TDataSet)
3288: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3289: Function DataSetSectionName( DataSet : TDataSet) : string
3290: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string)
3291: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3292: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
      Options: TLocateOptions) : Boolean
3293: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3294: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3295: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3296: Function ConfirmDelete : Boolean
3297: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3298: Procedure CheckRequiredField( Field : TField)
3299: Procedure CheckRequiredFields( const Fields : array of TField)
3300: Function DateToSQL( Value : TDateTime) : string
3301: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3302: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3303: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
      HighEmpty:Double;Inclusive:Bool):string
3304: Function StrMaskSQL( const Value : string) : string
3305: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3306: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3307: Procedure _DBError( const Msg : string)
3308:  Const('TrueExpr','String '0=0
3309:  Const('sdfStandard16','String '''''''mm''/''dd''/''yyyy''''''
3310:  Const('sdfStandard32','String ''''''''dd/mm/yyyy''''''''
3311:  Const('sdfOracle','String '"TO_DATE('''"dd/mm/yyyy"''', ''DD/MM/YYYY'')"
3312:  Const('sdfInterbase','String '"CAST('''"mm"/"dd"/"yyyy"'' AS DATE)"
3313:  Const('sdfMSSQL','String '"CONVERT(datetime, '''"mm"/"dd"/"yyyy"''', 103)"
```

```
3314:  AddTypeS('Largeint', 'Longint
3315:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3316:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3317:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3318:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3319:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3320: (*----------------------------------------------------------------------------*)
3321: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3322: begin
3323:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3324:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3325:  Function JIniReadString( const FileName, Section, Line : string) : string
3326:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3327:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3328:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3329:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3330:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3331: end;
3332:
3333: (* === compile-time registration functions === *)
3334: (*----------------------------------------------------------------------------*)
3335: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3336: begin
3337:  'UnixTimeStart','LongInt'( 25569);
3338:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3339:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3340:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3341:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3342:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3343:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3344:  Function DayOfDate( const DateTime : TDateTime) : Integer
3345:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3346:  Function YearOfDate( const DateTime : TDateTime) : Integer
3347:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3348:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3349:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3350:  Function HourOfTime( const DateTime : TDateTime) : Integer
3351:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3352:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3353:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3354:  Function IsISOLongYear( const Year : Word) : Boolean;
3355:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3356:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3357:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3358:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3359:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3360:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3361:  Function JIsLeapYear( const Year : Integer) : Boolean;
3362:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3363:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3364:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3365:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3366:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3367:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3368:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3369:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3370:  Function HoursToMSecs( Hours : Integer) : Integer
3371:  Function MinutesToMSecs( Minutes : Integer) : Integer
3372:  Function SecondsToMSecs( Seconds : Integer) : Integer
3373:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3374:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3375:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3376:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3377:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3378:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3379:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3380:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3381:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3382:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3383:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3384:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3385:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3386:  Function DosDateTimeToStr( DateTime : Integer) : string
3387:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3388:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3389:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3390:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3391:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3392:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3393:  Function FileTimeToStr( const FileTime : TFileTime) : string
3394:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3395:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3396:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3397:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3398:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3399:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3400:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3401:   TJclUnixTime32', 'Longword
3402:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
```

```
3403:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3404:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3405:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3406:  Function JNullStamp : TTimeStamp
3407:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3408:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3409:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3410:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3411:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3412:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3413:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3414:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3415:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3416:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3417:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3418:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3419:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3420:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3421:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3422:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3423:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3424:   FindClass('TOBJECT'),'EJclDateTimeError
3425: end;
3426:
3427: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3428: begin
3429:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3430:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3431:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3432:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3433:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3434:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3435:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3436:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3437:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3438:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3439:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3440:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3441:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3442:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3443:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3444:  Function AbortShutDown : Boolean;
3445:  Function AbortShutDown1( const MachineName : string) : Boolean;
3446:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3447:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3448:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3449:   FindClass('TOBJECT'),'EJclCreateProcessError
3450:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3451:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
      Environment:PChar);
3452:   // with Add(EJclCreateProcessError) do
3453:  end;
3454:
3455:
3456: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3457: begin
3458:   //'AnsiSigns','Set').SetSet(['-', '+']);
3459:  'C1_UPPER','LongWord( $0001);
3460:  'C1_LOWER','LongWord( $0002);
3461:  'C1_DIGIT','LongWord').SetUInt( $0004);
3462:  'C1_SPACE','LongWord').SetUInt( $0008);
3463:  'C1_PUNCT','LongWord').SetUInt( $0010);
3464:  'C1_CNTRL','LongWord').SetUInt( $0020);
3465:  'C1_BLANK','LongWord').SetUInt( $0040);
3466:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3467:  'C1_ALPHA','LongWord').SetUInt( $0100);
3468:   AnsiChar', 'Char
3469:  Function StrIsAlpha( const S : AnsiString) : Boolean
3470:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3471:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3472:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3473:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3474:  Function StrIsDigit( const S : AnsiString) : Boolean
3475:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3476:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3477:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3478:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3479:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3480:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3481:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3482:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3483:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3484:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3485:  Function StrEscapedToString( const S : AnsiString) : AnsiString
3486:  Function JStrLower( const S : AnsiString) : AnsiString
3487:  Procedure StrLowerInPlace( var S : AnsiString)
3488:  ///Procedure StrLowerBuff( S : PAnsiChar)
3489:  Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3490:  Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
```

```
3491:  Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3492:  Function StrProper( const S : AnsiString) : AnsiString
3493:  //Procedure StrProperBuff( S : PAnsiChar)
3494:  Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3495:  Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3496:  Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3497:  Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3498:  Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3499:  Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3500:  Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3501:  Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3502:  Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3503:  Function StrReverse( const S : AnsiString) : AnsiString
3504:  Procedure StrReverseInPlace( var S : AnsiString)
3505:  Function StrSingleQuote( const S : AnsiString) : AnsiString
3506:  Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
3507:  Function StrStringToEscaped( const S : AnsiString) : AnsiString
3508:  Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3509:  Function StrToHex( const Source : AnsiString) : AnsiString
3510:  Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3511:  Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3512:  Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3513:  Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3514:  Function StrTrimQuotes( const S : AnsiString) : AnsiString
3515:  Function JStrUpper( const S : AnsiString) : AnsiString
3516:  Procedure StrUpperInPlace( var S : AnsiString)
3517:  //Procedure StrUpperBuff( S : PAnsiChar)
3518:  Function StrOemToAnsi( const S : AnsiString) : AnsiString
3519:  Function StrAnsiToOem( const S : AnsiString) : AnsiString
3520:  Procedure StrAddRef( var S : AnsiString)
3521:  Function StrAllocSize( const S : AnsiString) : Longint
3522:  Procedure StrDecRef( var S : AnsiString)
3523:  //Function StrLen( S : PAnsiChar) : Integer
3524:  Function StrLength( const S : AnsiString) : Longint
3525:  Function StrRefCount( const S : AnsiString) : Longint
3526:  Procedure StrResetLength( var S : AnsiString)
3527:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3528:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3529:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3530:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3531:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3532:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3533:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3534:  Function StrFillChar(const C: Char; Count: Integer): string)');
3535:  Function IntFillChar(const I: Integer; Count: Integer): string)');
3536:  Function ByteFillChar(const B: Byte; Count: Integer): string)');
3537:  Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;');
3538:  Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3539:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3540:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3541:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3542:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
3543:  Function StrIPos( const SubStr, S : AnsiString) : Integer
3544:  Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3545:  Function StrLastPos( const SubStr, S : AnsiString) : Integer
3546:  Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3547:  Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3548:  Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3549:  Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3550:  Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3551:  Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3552:  //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3553:  //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3554:  Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3555:  Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3556:  Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3557:  Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
3558:  Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3559:  Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3560:  Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3561:  Function CharIsAlpha( const C : AnsiChar) : Boolean
3562:  Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3563:  Function CharIsBlank( const C : AnsiChar) : Boolean
3564:  Function CharIsControl( const C : AnsiChar) : Boolean
3565:  Function CharIsDelete( const C : AnsiChar) : Boolean
3566:  Function CharIsDigit( const C : AnsiChar) : Boolean
3567:  Function CharIsLower( const C : AnsiChar) : Boolean
3568:  Function CharIsNumberChar( const C : AnsiChar) : Boolean
3569:  Function CharIsPrintable( const C : AnsiChar) : Boolean
3570:  Function CharIsPunctuation( const C : AnsiChar) : Boolean
3571:  Function CharIsReturn( const C : AnsiChar) : Boolean
3572:  Function CharIsSpace( const C : AnsiChar) : Boolean
3573:  Function CharIsUpper( const C : AnsiChar) : Boolean
3574:  Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3575:  Function CharType( const C : AnsiChar) : Word
3576:  Function CharHex( const C : AnsiChar) : Byte
3577:  Function CharLower( const C : AnsiChar) : AnsiChar
3578:  Function CharUpper( const C : AnsiChar) : AnsiChar
3579:  Function CharToggleCase( const C : AnsiChar) : AnsiChar
```

```
3580:  Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3581:  Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3582:  Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer) : Integer
3583:  Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar) : Integer
3584:  Procedure StrIToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3585:  Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3586:  Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3587:  Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)
3588:  Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)
3589:  Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)
3590:  Function AddStringToStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3591:  Function BooleanToStr( B : Boolean) : AnsiString
3592:  Function FileToString( const FileName : AnsiString) : AnsiString
3593:  Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean)
3594:  Function StrToken( var S : AnsiString; Separator : AnsiChar) : AnsiString
3595:  Procedure StrTokens( const S : AnsiString; const List : TStrings)
3596:  Procedure StrTokenToStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings)
3597:  //Function StrWord( var S : PAnsiChar; out Word : AnsiString) : Boolean
3598:  Function StrToFloatSafe( const S : AnsiString) : Float
3599:  Function StrToIntSafe( const S : AnsiString) : Integer
3600:  Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer);
3601:  Function ArrayOf( List : TStrings) : TDynStringArray;
3602:   EJclStringError', 'EJclError
3603:  function IsClass(Address: TObject): Boolean;
3604:  function IsObject(Address: TObject): Boolean;
3605:  // Console Utilities
3606:  //function ReadKey: Char;
3607:  function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3608:  function JclGUIDToString(const GUID: TGUID): string;
3609:  function JclStringToGUID(const S: string): TGUID;
3610: end;
3611:
3612:
3613: ****************************************** uPSI_JvDBUtil;
3614: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const
      Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3615: Function GetQueryResult( const DatabaseName, SQL : string) : Variant
3616: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant;
      const AResultName : string) : Variant
3617: //Function StrFieldDesc( Field : FLDDesc) : string
3618: Function Var2Type( V : Variant; const VarType : Integer) : Variant
3619: Procedure CopyRecord( DataSet : TDataSet)
3620: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string;
      MasterField : Word; ModOp, DelOp : RINTQual)
3621: Procedure AddMasterPassword( Table : TTable; pswd : string)
3622: Procedure PackTable( Table : TTable)
3623: Procedure PackEncryptedTable( Table : TTable; pswd : string)
3624: Function EncodeQuotes( const S : string) : string
3625: Function Cmp( const S1, S2 : string) : Boolean
3626: Function SubStr( const S : string; const Index : Integer; const Separator : string) : string
3627: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string) : string
3628: Function ReplaceString( S : string; const OldPattern, NewPattern : string) : string
3629: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer)
3630: ******************************************uPSI_JvJvBDEUtils;***************
3631: //JvBDEUtils
3632: Function CreateDbLocate : TJvLocateObject
3633:  //Function CheckOpen( Status : DBIResult) : Boolean
3634: Procedure FetchAllRecords( DataSet : TBDEDataSet)
3635: Function TransActive( Database : TDatabase) : Boolean
3636: Function AsyncQrySupported( Database : TDatabase) : Boolean
3637: Function GetQuoteChar( Database : TDatabase) : string
3638: Procedure ExecuteQuery( const DbName, QueryText : string)
3639: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string)
3640: Function FieldLogicMap( FldType : TFieldType) : Integer
3641: Function FieldSubtypeMap( FldType : TFieldType) : Integer Value : string; Buffer : Pointer)
3642: Function GetAliasPath( const AliasName : string) : string
3643: Function IsDirectory( const DatabaseName : string) : Boolean
3644: Function GetBdeDirectory : string
3645: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean
3646: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3647: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3648: Function DataSetRecNo( DataSet : TDataSet) : Longint
3649: Function DataSetRecordCount( DataSet : TDataSet) : Longint
3650: Function DataSetPositionStr( DataSet : TDataSet) : string
3651: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)
3652: Function CurrentRecordDeleted( DataSet : TBDEDataSet) : Boolean
3653: Function IsFilterApplicable( DataSet : TDataSet) : Boolean
3654: Function IsBookmarkStable( DataSet : TBDEDataSet) : Boolean
3655: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3656: Procedure RestoreIndex( Table : TTable)
3657: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3658: Procedure PackTable( Table : TTable)
3659: Procedure ReindexTable( Table : TTable)
3660: Procedure BdeFlushBuffers
3661: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer
3662: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3663: Procedure DbNotSupported
3664: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
```

```
3665: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3666: Procedure
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3667: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3668: ******************************************uPSI_JvDateUtil;
3669: function CurrentYear: Word;
3670: function IsLeapYear(AYear: Integer): Boolean;
3671: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3672: function FirstDayOfPrevMonth: TDateTime;
3673: function LastDayOfPrevMonth: TDateTime;
3674: function FirstDayOfNextMonth: TDateTime;
3675: function ExtractDay(ADate: TDateTime): Word;
3676: function ExtractMonth(ADate: TDateTime): Word;
3677: function ExtractYear(ADate: TDateTime): Word;
3678: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3679: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3680: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3681: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3682: function ValidDate(ADate: TDateTime): Boolean;
3683: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3684: function MonthsBetween(Date1, Date2: TDateTime): Double;
3685: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3686: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3687: function DaysBetween(Date1, Date2: TDateTime): Longint;
3688: { The same as previous but if Date2 < Date1 result = 0 }
3689: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3690: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3691: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3692: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3693: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3694: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3695: { String to date conversions }
3696: function GetDateOrder(const DateFormat: string): TDateOrder;
3697: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3698: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3699: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3700: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3701: function DefDateFormat(FourDigitYear: Boolean): string;
3702: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3703: ------------------------------------------------------------------------------
3704: ******************************** JvUtils;*****************************
3705: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3706: function GetWordOnPos(const S: string; const P: Integer): string;
3707: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3708: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3709: { SubStr returns substring from string, S, separated with Separator string}
3710: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3711: { SubStrEnd same to previous function but Index numerated from the end of string }
3712: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3713: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3714: function SubWord(P: PChar; var P2: PChar): string;
3715: { NumberByWord returns the text representation of
3716:   the number, N, in normal russian language. Was typed from Monitor magazine }
3717: function NumberByWord(const N: Longint): string;
3718: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3719:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3720: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3721: { GetXYByPos is same to previous function, but returns X position in line too}
3722: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3723: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3724: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3725: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3726: function ConcatSep(const S, S2, Separator: string): string;
3727: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3728: function ConcatLeftSep(const S, S2, Separator: string): string;
3729: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3730: function MinimizeString(const S: string; const MaxLen: Integer): string;
3731: { Next 4 function for russian chars transliterating.
3732:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3733: procedure Dos2Win(var S: string);
3734: procedure Win2Dos(var S: string);
3735: function Dos2WinRes(const S: string): string;
3736: function Win2DosRes(const S: string): string;
3737: function Win2Koi(const S: string): string;
3738: { Spaces returns string consists on N space chars }
3739: function Spaces(const N: Integer): string;
3740: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3741: function AddSpaces(const S: string; const N: Integer): string;
3742: { function LastDate for russian users only } { returns date relative to current date: '' }
3743: function LastDate(const Dat: TDateTime): string;
3744: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3745: function CurrencyToStr(const Cur: currency): string;
3746: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3747: function Cmp(const S1, S2: string): Boolean;
3748: { StringCat add S2 string to S1 and returns this string }
3749: function StringCat(var S1: string; S2: string): string;
3750: { HasChar returns True, if Char, Ch, contains in string, S }
3751: function HasChar(const Ch: Char; const S: string): Boolean;
```

```
3752: function HasAnyChar(const Chars: string; const S: string): Boolean;
3753: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3754: function CountOfChar(const Ch: Char; const S: string): Integer;
3755: function DefStr(const S: string; Default: string): string;
3756: {**** files routines}
3757: { GetWinDir returns Windows folder name }
3758: function GetWinDir: TFileName;
3759: function GetSysDir: String;
3760: { GetTempDir returns Windows temporary folder name }
3761: function GetTempDir: string;
3762: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3763: function GenTempFileName(FileName: string): string;
3764: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3765: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3766: { ClearDir clears folder Dir }
3767: function ClearDir(const Dir: string): Boolean;
3768: { DeleteDir clears and than delete folder Dir }
3769: function DeleteDir(const Dir: string): Boolean;
3770: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3771: function FileEquMask(FileName, Mask: TFileName): Boolean;
3772: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3773:   Masks must be separated with comma (';') }
3774: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3775: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3776: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3777: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3778: { FileGetInfo fills SearchRec record for specified file attributes}
3779: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3780: { HasSubFolder returns True, if folder APath contains other folders }
3781: function HasSubFolder(APath: TFileName): Boolean;
3782: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3783: function IsEmptyFolder(APath: TFileName): Boolean;
3784: { AddSlash add slash Char to Dir parameter, if needed }
3785: procedure AddSlash(var Dir: TFileName);
3786: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3787: function AddSlash2(const Dir: TFileName): string;
3788: { AddPath returns FileName with Path, if FileName not contain any path }
3789: function AddPath(const FileName, Path: TFileName): TFileName;
3790: function AddPaths(const PathList, Path: string): string;
3791: function ParentPath(const Path: TFileName): TFileName;
3792: function FindInPath(const FileName, PathList: string): TFileName;
3793: function FindInPaths(const fileName,paths: String): String;
3794: {$IFNDEF BCB1}
3795: { BrowseForFolder displays Browse For Folder dialog }
3796: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3797: {$ENDIF BCB1}
3798: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3799: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3800: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3801: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3802:
3803: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3804: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3805: { HasParam returns True, if program running with specified parameter, Param }
3806: function HasParam(const Param: string): Boolean;
3807: function HasSwitch(const Param: string): Boolean;
3808: function Switch(const Param: string): string;
3809: { ExePath returns ExtractFilePath(ParamStr(0)) }
3810: function ExePath: TFileName;
3811: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3812: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3813: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3814: {**** Graphic routines }
3815: { TTFontSelected returns True, if True Type font is selected in specified device context }
3816: function TTFontSelected(const DC: HDC): Boolean;
3817: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3818: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3819: {**** Windows routines }
3820: { SetWindowTop put window to top without recreating window }
3821: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3822: {**** other routines }
3823: { KeyPressed returns True, if Key VK is now pressed }
3824: function KeyPressed(VK: Integer): Boolean;
3825: procedure SwapInt(var Int1, Int2: Integer);
3826: function IntPower(Base, Exponent: Integer): Integer;
3827: function ChangeTopException(E: TObject): TObject;
3828: function StrToBool(const S: string): Boolean;
3829: {$IFNDEF COMPILER3_UP}
3830: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3831:   Length of MaxLen bytes. The compare operation is controlled by the
3832:   current Windows locale. The return value is the same as for CompareStr. }
3833: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3834: function AnsiStrIComp(S1, S2: PChar): Integer;
3835: {$ENDIF}
3836: function Var2Type(V: Variant; const VarType: Integer): Variant;
3837: function VarToInt(V: Variant): Integer;
3838: function VarToFloat(V: Variant): Double;
```

```
3839: { following functions are not documented because they are don't work properly , so don't use them }
3840: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3841: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3842: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3843: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3844: function GetParameter: string;
3845: function GetLongFileName(FileName: string): string;
3846: {* from  FileCtrl}
3847: function DirectoryExists(const Name: string): Boolean;
3848: procedure ForceDirectories(Dir: string);
3849: {# from  FileCtrl}
3850: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3851: function GetComputerID: string;
3852: function GetComputerName: string;
3853: {**** string routines }
3854: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 function }
3855: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3856: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3857:   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 }
3858: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3859: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3860: function CountOfLines(const S: string): Integer;
3861: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3862: procedure DeleteEmptyLines(Ss: TStrings);
3863: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3864:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3865: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3866: {**** files routines - }
3867: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3868:   Resource can be compressed using MS Compress program}
3869: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3870: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3871: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3872: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3873: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3874: { IniReadSection read section, Section, from ini-file,
3875:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3876:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3877: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3878: { LoadTextFile load text file, FileName, into string }
3879: function LoadTextFile(const FileName: TFileName): string;
3880: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3881: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3882: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3883: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3884: {$IFDEF COMPILER3_UP}
3885: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3886: function TargetFileName(const FileName: TFileName): TFileName;
3887: { return filename ShortCut linked to }
3888: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3889: {$ENDIF COMPILER3_UP}
3890: {**** Graphic routines - }
3891: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3892: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3893: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3894: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3895: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3896: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3897: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3898: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3899: { Cinema draws some visual effect }
3900: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3901: { Roughed fills rect with special 3D pattern }
3902: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3903: { 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 }
3904: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3905: { TextWidth calculate text with for writing using standard desktop font }
3906: function TextWidth(AStr: string): Integer;
3907: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3908: function DefineCursor(Identifer: PChar): TCursor;
3909: {**** other routines - }
3910: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3911: function FindFormByClass(FormClass: TFormClass): TForm;
3912: function FindFormByClassName(FormClassName: string): TForm;
3913: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3914:   having Tag property value, equaled to Tag parameter }
3915: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3916: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3917: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3918: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3919: function RBTag(Parent: TWinControl): Integer;
3920: { AppMinimized returns True, if Application is minimized }
3921: function AppMinimized: Boolean;
3922: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3923:   if Caption parameter = '', it replaced with Application.Title }
3924: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
```

```
3925: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3926:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3927: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3928:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3929: { Delay stop program execution to MSec msec }
3930: procedure Delay(MSec: Longword);
3931: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3932: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3933: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3934: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3935: function PanelBorder(Panel: TCustomPanel): Integer;
3936: function Pixels(Control: TControl; APixels: Integer): Integer;
3937: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3938: procedure Error(const Msg: string);
3939: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3940:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3941:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3942: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3943:   const HideSelColor: Boolean): string;
3944: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3945:   const HideSelColor: Boolean): Integer;
3946: function ItemHtPlain(const Text: string): string;
3947: { ClearList - clears list of TObject }
3948: procedure ClearList(List: TList);
3949: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3950: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3951: { RTTI support }
3952: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3953: function GetPropStr(Obj: TObject; const PropName: string): string;
3954: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3955: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3956: procedure PrepareIniSection(SS: TStrings);
3957: { following functions are not documented because they are don't work properly, so don't use them }
3958: {$IFDEF COMPILER2}
3959: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3960: {$ENDIF}
3961:
3962: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3963: begin
3964:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3965:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3966:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
3967:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3968:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3969:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3970:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3971:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3972: end;
3973:
3974: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3975: begin
3976:  Const('MaxInitStrNum','LongInt'( 9);
3977: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
      : array of AnsiString; MaxSplit : Integer) : Integer
3978: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
      string; MaxSplit : Integer) : Integer
3979: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
      QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3980: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3981: Function JvStrStrip( S : string) : string
3982: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3983: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3984: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3985: Function StrEatWhiteSpace( const S : string) : string
3986: Function HexToAscii( const S : AnsiString) : AnsiString
3987: Function AsciiToHex( const S : AnsiString) : AnsiString
3988: Function StripQuotes( const S1 : AnsiString) : AnsiString
3989: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3990: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3991: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3992: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3993: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3994: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
3995: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
3996: Function JvValidIdentifier( S1 : String) : Boolean
3997: Function JvEndChar( X : AnsiChar) : Boolean
3998: Procedure JvGetToken( S1, S2 : PAnsiChar)
3999: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
4000: Function IsKeyword( S1 : PAnsiChar) : Boolean
4001: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
4002: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
4003: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
4004: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
4005: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
4006: Function GetTokenCount : Integer
4007: Procedure ResetTokenCount
4008: end;
4009:
```

```
4010: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
4011: begin
4012:   SIRegister_TJvQueryParamsDialog(CL);
4013:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
4014: end;
4015:
4016: ******************************** JvStrUtil /  JvStrUtils;****************************
4017: function FindNotBlankCharPos(const S: string): Integer;
4018: function AnsiChangeCase(const S: string): string;
4019: function GetWordOnPos(const S: string; const P: Integer): string;
4020: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4021: function Cmp(const S1, S2: string): Boolean;
4022: { Spaces returns string consists on N space chars }
4023: function Spaces(const N: Integer): string;
4024: { HasChar returns True, if char, Ch, contains in string, S }
4025: function HasChar(const Ch: Char; const S: string): Boolean;
4026: function HasAnyChar(const Chars: string; const S: string): Boolean;
4027: { SubStr returns substring from string, S, separated with Separator string}
4028: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
4029: { SubStrEnd same to previous function but Index numerated from the end of string }
4030: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4031: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
4032: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
4033: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4034: { GetXYByPos is same to previous function, but returns X position in line too}
4035: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4036: { AddSlash returns string with added slash char to Dir parameter, if needed }
4037: function AddSlash2(const Dir: TFileName): string;
4038: { AddPath returns FileName with Path, if FileName not contain any path }
4039: function AddPath(const FileName, Path: TFileName): TFileName;
4040: { ExePath returns ExtractFilePath(ParamStr(0)) }
4041: function ExePath: TFileName;
4042: function LoadTextFile(const FileName: TFileName): string;
4043: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4044: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4045: function ConcatSep(const S, S2, Separator: string): string;
4046: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4047: function FileEquMask(FileName, Mask: TFileName): Boolean;
4048: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4049:   Masks must be separated with comma (';') }
4050: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4051: function StringEndsWith(const Str, SubStr: string): Boolean;
4052: function ExtractFilePath2(const FileName: string): string;
4053: function StrToOem(const AnsiStr: string): string;
4054: { StrToOem translates a string from the Windows character set into the OEM character set. }
4055: function OemToAnsiStr(const OemStr: string): string;
4056: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4057: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4058: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4059: function ReplaceStr(const S, Srch, Replace: string): string;
4060: { Returns string with every occurrence of Srch string replaced with Replace string. }
4061: function DelSpace(const S: string): string;
4062: { DelSpace return a string with all white spaces removed. }
4063: function DelChars(const S: string; Chr: Char): string;
4064: { DelChars return a string with all Chr characters removed. }
4065: function DelBSpace(const S: string): string;
4066: { DelBSpace trims leading spaces from the given string. }
4067: function DelESpace(const S: string): string;
4068: { DelESpace trims trailing spaces from the given string. }
4069: function DelRSpace(const S: string): string;
4070: { DelRSpace trims leading and trailing spaces from the given string. }
4071: function DelSpace1(const S: string): string;
4072: { DelSpace1 return a string with all non-single white spaces removed. }
4073: function Tab2Space(const S: string; Numb: Byte): string;
4074: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4075: function NPos(const C: string; S: string; N: Integer): Integer;
4076: { NPos searches for a N-th position of substring C in a given string. }
4077: function MakeStr(C: Char; N: Integer): string;
4078: function MS(C: Char; N: Integer): string;
4079: { MakeStr return a string of length N filled with character C. }
4080: function AddChar(C: Char; const S: string; N: Integer): string;
4081: { AddChar return a string left-padded to length N with characters C. }
4082: function AddCharR(C: Char; const S: string; N: Integer): string;
4083: { AddCharR return a string right-padded to length N with characters C. }
4084: function LeftStr(const S: string; N: Integer): string;
4085: { LeftStr return a string right-padded to length N with blanks. }
4086: function RightStr(const S: string; N: Integer): string;
4087: { RightStr return a string left-padded to length N with blanks. }
4088: function CenterStr(const S: string; Len: Integer): string;
4089: { CenterStr centers the characters in the string based upon the Len specified. }
4090: function CompStr(const S1, S2: string): Integer;
4091: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4092: function CompText(const S1, S2: string): Integer;
4093: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4094: function Copy2Symb(const S: string; Symb: Char): string;
4095: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4096: function Copy2SymbDel(var S: string; Symb: Char): string;
4097: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4098: function Copy2Space(const S: string): string;
```

```
4099: { Copy2Symb returns a substring of a string S from begining to first white space. }
4100: function Copy2SpaceDel(var S: string): string;
4101: { Copy2SpaceDel returns a substring of a string S from begining to first
4102:   white space and removes this substring from S. }
4103: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4104: { Returns string, with the first letter of each word in uppercase,
4105:   all other letters in lowercase. Words are delimited by WordDelims. }
4106: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4107: { WordCount given a set of word delimiters, returns number of words in S. }
4108: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4109: { Given a set of word delimiters, returns start position of N'th word in S. }
4110: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4111: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4112: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4113: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4114:   delimiters, return the N'th word in S. }
4115: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4116: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4117: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4118: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4119: function QuotedString(const S: string; Quote: Char): string;
4120: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4121: function ExtractQuotedString(const S: string; Quote: Char): string;
4122: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4123:    and reduces pairs of Quote characters within the quoted string to a single character. }
4124: function FindPart(const HelpWilds, InputStr: string): Integer;
4125: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4126: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4127: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4128: function XorString(const Key, Src: ShortString): ShortString;
4129: function XorEncode(const Key, Source: string): string;
4130: function XorDecode(const Key, Source: string): string;
4131: { ** Command line routines ** }
4132: {$IFNDEF COMPILER4_UP}
4133: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4134: {$ENDIF}
4135: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4136: { ** Numeric string handling routines ** }
4137: function Numb2USA(const S: string): string;
4138: { Numb2USA converts numeric string S to USA-format. }
4139: function Dec2Hex(N: Longint; A: Byte): string;
4140: function D2H(N: Longint; A: Byte): string;
4141: { Dec2Hex converts the given value to a hexadecimal string representation
4142:   with the minimum number of digits (A) specified. }
4143: function Hex2Dec(const S: string): Longint;
4144: function H2D(const S: string): Longint;
4145: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4146: function Dec2Numb(N: Longint; A, B: Byte): string;
4147: { Dec2Numb converts the given value to a string representation with the
4148:   base equal to B and with the minimum number of digits (A) specified. }
4149: function Numb2Dec(S: string; B: Byte): Longint;
4150: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4151: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4152: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4153: function IntToRoman(Value: Longint): string;
4154: { IntToRoman converts the given value to a roman numeric string representation. }
4155: function RomanToInt(const S: string): Longint;
4156: { RomanToInt converts the given string to an integer value. If the string
4157:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4158: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4159: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4160: ********************************** JvFileUtil;********************************
4161: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4162: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4163: procedure MoveFile(const FileName, DestName: TFileName);
4164: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4165: {$IFDEF COMPILER4_UP}
4166: function GetFileSize(const FileName: string): Int64;
4167: {$ELSE}
4168: function GetFileSize(const FileName: string): Longint;
4169: {$ENDIF}
4170: function FileDateTime(const FileName: string): TDateTime;
4171: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4172: function DeleteFiles(const FileMask: string): Boolean;
4173: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4174: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4175: function NormalDir(const DirName: string): string;
4176: function RemoveBackSlash(const DirName: string): string;
4177: function ValidFileName(const FileName: string): Boolean;
4178: function DirExists(Name: string): Boolean;
4179: procedure ForceDirectories(Dir: string);
4180: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4181:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4182: {$IFDEF COMPILER4_UP}
4183: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4184: {$ENDIF}
4185: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4186: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
```

```
4187: {$IFDEF COMPILER4_UP}
4188: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4189: {$ENDIF}
4190: function GetTempDir: string;
4191: function GetWindowsDir: string;
4192: function GetSystemDir: string;
4193: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4194: {$IFDEF WIN32}
4195: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4196: function ShortToLongFileName(const ShortName: string): string;
4197: function ShortToLongPath(const ShortName: string): string;
4198: function LongToShortFileName(const LongName: string): string;
4199: function LongToShortPath(const LongName: string): string;
4200: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4201: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4202: {$ENDIF WIN32}
4203: {$IFNDEF COMPILER3_UP}
4204: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4205: {$ENDIF}
4206: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4207: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4208: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4209: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4210:
4211: //**************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4212:  Procedure VariantClear( var V : Variant)
4213:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4214:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4215:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4216:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4217:  Procedure VariantSub( const src : Variant; var dst : Variant)
4218:  Procedure VariantMul( const src : Variant; var dst : Variant)
4219:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4220:  Procedure VariantMod( const src : Variant; var dst : Variant)
4221:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4222:  Procedure VariantOr( const src : Variant; var dst : Variant)
4223:  Procedure VariantXor( const src : Variant; var dst : Variant)
4224:  Procedure VariantShl( const src : Variant; var dst : Variant)
4225:  Procedure VariantShr( const src : Variant; var dst : Variant)
4226:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4227:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4228:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4229:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4230:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4231:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4232:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4233:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4234:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4235:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4236:  Function VariantNot( const V1 : Variant) : Variant
4237:  Function VariantNeg( const V1 : Variant) : Variant
4238:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4239:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4240:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4241:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4242:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4243:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4244:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4245:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4246:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4247:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4248: end;
4249:
4250: ************************************unit uPSI_JvgUtils;************************************
4251: function IsEven(I: Integer): Boolean;
4252: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4253: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4254: procedure SwapInt(var I1, I2: Integer);
4255: function Spaces(Count: Integer): string;
4256: function DupStr(const Str: string; Count: Integer): string;
4257: function DupChar(C: Char; Count: Integer): string;
4258: procedure Msg(const AMsg: string);
4259: function RectW(R: TRect): Integer;
4260: function RectH(R: TRect): Integer;
4261: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4262: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4263: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4264: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4265:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4266: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4267: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4268:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
     TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4269: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4270: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4271:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4272: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4273: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4274: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
```

```
4275:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4276:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4277:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4278: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4279:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4280:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4281:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4282: procedure BringParentWindowToTop(Wnd: TWinControl);
4283: function GetParentForm(Control: TControl): TForm;
4284: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4285: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4286: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4287: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4288: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4289: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4290: function CalcMathString(AExpression: string): Single;
4291: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4292: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4293: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4294: procedure TypeStringOnKeyboard(const S: string);
4295: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4296: procedure DrawTextExtAligned(Canvas:TCanvas;const
     Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4297: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4298: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4299: function ComponentToString(Component: TComponent): string;
4300: procedure StringToComponent(Component: TComponent; const Value: string);
4301: function PlayWaveResource(const ResName: string): Boolean;
4302: function UserName: string;
4303: function ComputerName: string;
4304: function CreateIniFileName: string;
4305: function ExpandString(const Str: string; Len: Integer): string;
4306: function Transliterate(const Str: string; RusToLat: Boolean): string;
4307: function IsSmallFonts: Boolean;
4308: function SystemColorDepth: Integer;
4309: function GetFileTypeJ(const FileName: string): TglFileType;
4310: Function GetFileType( hFile : THandle) : DWORD');
4311: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4312: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4313:
4314: { ************************************Utility routines of unit classes}
4315: function LineStart(Buffer, BufPos: PChar): PChar
4316: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4317:   'Strings: TStrings): Integer
4318:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4319:  Procedure RegisterClass( AClass : TPersistentClass)
4320:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4321:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4322:  Procedure UnRegisterClass( AClass : TPersistentClass)
4323:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4324:  Procedure UnRegisterModuleClasses( Module : HMODULE)
4325:  Function FindGlobalComponent( const Name : string) : TComponent
4326:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4327:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4328:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4329:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4330:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4331:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4332:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4333:  Procedure GlobalFixupReferences
4334:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4335:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4336:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4337:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4338:  Procedure RemoveFixups( Instance : TPersistent)
4339:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4340:  Procedure BeginGlobalLoading
4341:  Procedure NotifyGlobalLoading
4342:  Procedure EndGlobalLoading
4343:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4344:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4345:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4346:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4347:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
4348: // Function AllocateHWnd( Method : TWndMethod) : HWND
4349:  Procedure DeallocateHWnd( Wnd : HWND)
4350:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4351: ************************************unit uPSI_SqlTimSt and DB;*********************************
4352: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4353: Function VarSQLTimeStampCreate3: Variant;
4354: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4355: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4356: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4357: Function VarSQLTimeStamp : TVarType
4358: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4359: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4360: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4361: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4362: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
```

```
4363: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4364: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4365: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4366: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4367: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4368: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4369: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4370: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4371:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4372: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4373: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4374: Procedure DisposeMem( var Buffer, Size : Integer)
4375: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4376: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4377: Function VarTypeToDataType( VarType : Integer) : TFieldType
4378: ******************************************unit JvStrings;************************************
4379: {template functions}
4380: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4381: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4382: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4383: function RemoveMasterBlocks(const SourceStr: string): string;
4384: function RemoveFields(const SourceStr: string): string;
4385: {http functions}
4386: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4387: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4388: {set functions}
4389: procedure SplitSet(AText: string; AList: TStringList);
4390: function JoinSet(AList: TStringList): string;
4391: function FirstOfSet(const AText: string): string;
4392: function LastOfSet(const AText: string): string;
4393: function CountOfSet(const AText: string): Integer;
4394: function SetRotateRight(const AText: string): string;
4395: function SetRotateLeft(const AText: string): string;
4396: function SetPick(const AText: string; AIndex: Integer): string;
4397: function SetSort(const AText: string): string;
4398: function SetUnion(const Set1, Set2: string): string;
4399: function SetIntersect(const Set1, Set2: string): string;
4400: function SetExclude(const Set1, Set2: string): string;
4401: {replace any <,> etc by &lt; &gt;}
4402: function XMLSafe(const AText: string): string;
4403: {simple hash, Result can be used in Encrypt}
4404: function Hash(const AText: string): Integer;
4405: { Base64 encode and decode a string }
4406: function B64Encode(const S: AnsiString): AnsiString;
4407: function B64Decode(const S: AnsiString): AnsiString;
4408: {Basic encryption from a Borland Example}
4409: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4410: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4411: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4412: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4413: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4414: procedure CSVToTags(Src, Dst: TStringList);
4415: // converts a csv list to a tagged string list
4416: procedure TagsToCSV(Src, Dst: TStringList);
4417: // converts a tagged string list to a csv list
4418: // only fieldnames from the first record are scanned ib the other records
4419: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4420: {selects akey=avalue from Src and returns recordset in Dst}
4421: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4422: {filters Src for akey=avalue}
4423: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4424: {orders a tagged Src list by akey}
4425: function PosStr(const FindString, SourceString: string;
4426:   StartPos: Integer = 1): Integer;
4427: { PosStr searches the first occurrence of a substring FindString in a string
4428:   given by SourceString with case sensitivity (upper and lower case characters
4429:   are differed). This function returns the index value of the first character
4430:   of a specified substring from which it occurs in a given string starting with
4431:   StartPos character index. If a specified substring is not found Q_PosStr
4432:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4433: function PosStrLast(const FindString, SourceString: string): Integer;
4434: {finds the last occurance}
4435: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4436: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4437: { PosText searches the first occurrence of a substring FindString in a string
4438:   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
4439: function PosTextLast(const FindString, SourceString: string): Integer;
4440: {finds the last occurance}
4441: function NameValuesToXML(const AText: string): string;
4442: {$IFDEF MSWINDOWS}
4443: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4444: {$ENDIF MSWINDOWS}
4445: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4446: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4447: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4448: procedure SaveString(const AFile, AText: string);
4449: Procedure SaveStringasFile( const AFile, AText : string)
```

```
4450: function LoadStringJ(const AFile: string): string;
4451: Function LoadStringofFile( const AFile : string) : string
4452: Procedure SaveStringtoFile( const AFile, AText : string)
4453: Function LoadStringfromFile( const AFile : string) : string
4454: function HexToColor(const AText: string): TColor;
4455: function UppercaseHTMLTags(const AText: string): string;
4456: function LowercaseHTMLTags(const AText: string): string;
4457: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4458: function RelativePath(const ASrc, ADst: string): string;
4459: function GetToken(var Start: Integer; const SourceText: string): string;
4460: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4461: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4462: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4463: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4464: // parses the beginning of an attribute: space + alpha character
4465: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4466: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4467: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4468: // parses all name=value attributes to the attributes TStringList
4469: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4470: // checks if a name="value" pair exists and returns any value
4471: function GetStrValue(const AText, AName, ADefault: string): string;
4472: // retrieves string value from a line like:
4473: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4474: // returns ADefault when not found
4475: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4476: // same for a color
4477: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4478: // same for an Integer
4479: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4480: // same for a float
4481: function GetBoolValue(const AText, AName: string): Boolean;
4482: // same for Boolean but without default
4483: function GetValue(const AText, AName: string): string;
4484: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4485: procedure SetValue(var AText: string; const AName, AValue: string);
4486: // sets a string value in a line
4487: procedure DeleteValue(var AText: string; const AName: string);
4488: // deletes a AName="value" pair from AText
4489: procedure GetNames(AText: string; AList: TStringList);
4490: // get a list of names from a string with name="value" pairs
4491: function GetHTMLColor(AColor: TColor): string;
4492: // converts a color value to the HTML hex value
4493: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4494: // finds a string backward case sensitive
4495: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4496: // finds a string backward case insensitive
4497: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4498:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4499: // finds a text range, e.g. <TD>....</TD> case sensitive
4500: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4501:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4502: // finds a text range, e.g. <TD>....</td> case insensitive
4503: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4504:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4505: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4506: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4507:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4508: // finds a text range backward, e.g. <TD>....</td> case insensitive
4509: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4510:   var RangeEnd: Integer): Boolean;
4511: // finds a HTML or XML tag:  <....>
4512: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4513:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4514: // finds the innertext between opening and closing tags
4515: function Easter(NYear: Integer): TDateTime;
4516: // returns the easter date of a year.
4517: function GetWeekNumber(Today: TDateTime): string;
4518: //gets a datecode. Returns year and weeknumber in format: YYWW
4519: function ParseNumber(const S: string): Integer;
4520: // parse number returns the last position, starting from 1
4521: function ParseDate(const S: string): Integer;
4522: // parse a SQL style data string from positions 1,
4523: // starts and ends with #
4524:
4525: ****************************************unit JvJCLUtils;****************************************
4526:
4527: function VarIsInt(Value: Variant): Boolean;
4528:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4529: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4530: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4531: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4532: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4533: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4534: function GetWordOnPos(const S: string; const P: Integer): string;
4535: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4536: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4537: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4538: { GetWordOnPosEx working like GetWordOnPos function, but
```

```
4539:   also returns Word position in iBeg, iEnd variables }
4540: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4541: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4542: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4543: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4544: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4545: { GetEndPosCaret returns the caret position of the last char. For the position
4546:   after the last char of Text you must add 1 to the returned X value. }
4547: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
4548: { GetEndPosCaret returns the caret position of the last char. For the position
4549:   after the last char of Text you must add 1 to the returned X value. }
4550: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4551: function SubStrBySeparator(const S:string;const Index:Integer;const
      Separator:string;StartIndex:Int=1):string;
4552: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
      Separator:WideString;StartIndex:Int:WideString;
4553: { SubStrEnd same to previous function but Index numerated from the end of string }
4554: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4555: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4556: function SubWord(P: PChar; var P2: PChar): string;
4557: function CurrencyByWord(Value: Currency): string;
4558: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4559: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4560: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4561: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4562: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4563: { ReplaceString searches for all substrings, OldPattern,
4564:   in a string, S, and replaces them with NewPattern }
4565: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4566: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
      WideString;StartIndex:Integer=1):WideString;
4567: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4568: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4569: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4570: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4571:
4572: { Next 4 function for russian chars transliterating.
4573:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4574: procedure Dos2Win(var S: AnsiString);
4575: procedure Win2Dos(var S: AnsiString);
4576: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4577: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4578: function Win2Koi(const S: AnsiString): AnsiString;
4579: { FillString fills the string Buffer with Count Chars }
4580: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4581: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4582: { MoveString copies Count Chars from Source to Dest }
4583: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE} overload;
4584: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4585:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4586: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4587: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4588: { MoveWideChar copies Count WideChars from Source to Dest }
4589: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
      SUPPORTS_INLINE}
4590: { FillNativeChar fills Buffer with Count NativeChars }
4591: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4592: { MoveWideChar copies Count WideChars from Source to Dest }
4593: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4594: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4595: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4596: { Spaces returns string consists on N space chars }
4597: function Spaces(const N: Integer): string;
4598: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4599: function AddSpaces(const S: string; const N: Integer): string;
4600: function SpacesW(const N: Integer): WideString;
4601: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4602: { function LastDateRUS for russian users only }
4603: { returns date relative to current date: 'ääà äíý íàçàä' }
4604: function LastDateRUS(const Dat: TDateTime): string;
4605: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4606: function CurrencyToStr(const Cur: Currency): string;
4607: { HasChar returns True, if Char, Ch, contains in string, S }
4608: function HasChar(const Ch: Char; const S: string): Boolean;
4609: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4610: function HasAnyChar(const Chars: string; const S: string): Boolean;
4611: {$IFNDEF COMPILER12_UP}
4612: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4613: {$ENDIF ~COMPILER12_UP}
4614: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
      SUPPORTS_INLINE}
4615: function CountOfChar(const Ch: Char; const S: string): Integer;
4616: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
```

```
4617: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4618: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4619: function StrPosW(S, SubStr: PWideChar): PWideChar;
4620: function StrLenW(S: PWideChar): Integer;
4621: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4622: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4623: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4624: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4625: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4626:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4627: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4628: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4629: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4630: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4631: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
4632: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4633: Function ScreenPixelFormat : TPixelFormat
4634: Function ScreenColorCount : Integer
4635: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4636: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4637: //  SIRegister_TJvGradient(CL);
4638:
4639: {************************************* files routines}
4640: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4641: const
4642:   {$IFDEF MSWINDOWS}
4643:   DefaultCaseSensitivity = False;
4644:   {$ENDIF MSWINDOWS}
4645:   {$IFDEF UNIX}
4646:   DefaultCaseSensitivity = True;
4647:   {$ENDIF UNIX}
4648: { GenTempFileName returns temporary file name on
4649:   drive, there FileName is placed }
4650: function GenTempFileName(FileName: string): string;
4651: { GenTempFileNameExt same to previous function, but
4652:   returning filename has given extension, FileExt }
4653: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4654: { ClearDir clears folder Dir }
4655: function ClearDir(const Dir: string): Boolean;
4656: { DeleteDir clears and than delete folder Dir }
4657: function DeleteDir(const Dir: string): Boolean;
4658: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4659: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4660: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4661:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4662: function FileEquMasks(FileName, Masks: TFileName;
4663:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4664: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4665: {$IFDEF MSWINDOWS}
4666: { LZFileExpand expand file, FileSource,
4667:   into FileDest. Given file must be compressed, using MS Compress program }
4668: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4669: {$ENDIF MSWINDOWS}
4670: { FileGetInfo fills SearchRec record for specified file attributes}
4671: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4672: { HasSubFolder returns True, if folder APath contains other folders }
4673: function HasSubFolder(APath: TFileName): Boolean;
4674: { IsEmptyFolder returns True, if there are no files or
4675:   folders in given folder, APath}
4676: function IsEmptyFolder(APath: TFileName): Boolean;
4677: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4678: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4679: { AddPath returns FileName with Path, if FileName not contain any path }
4680: function AddPath(const FileName, Path: TFileName): TFileName;
4681: function AddPaths(const PathList, Path: string): string;
4682: function ParentPath(const Path: TFileName): TFileName;
4683: function FindInPath(const FileName, PathList: string): TFileName;
4684: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4685: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4686: { HasParam returns True, if program running with specified parameter, Param }
4687: function HasParam(const Param: string): Boolean;
4688: function HasSwitch(const Param: string): Boolean;
4689: function Switch(const Param: string): string;
4690: { ExePath returns ExtractFilePath(ParamStr(0)) }
4691: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4692: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4693: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4694: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4695: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4696: {**** Graphic routines }
4697: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4698: function IsTTFontSelected(const DC: HDC): Boolean;
4699: function KeyPressed(VK: Integer): Boolean;
4700: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4701: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4702: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4703: {**** Color routines }
4704: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
```

```
4705: function RGBToBGR(Value: Cardinal): Cardinal;
4706: //function ColorToPrettyName(Value: TColor): string;
4707: //function PrettyNameToColor(const Value: string): TColor;
4708: {**** other routines }
4709: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4710: function IntPower(Base, Exponent: Integer): Integer;
4711: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4712: function StrToBool(const S: string): Boolean;
4713: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4714: function VarToInt(V: Variant): Integer;
4715: function VarToFloat(V: Variant): Double;
4716: { following functions are not documented because they not work properly sometimes, so do not use them }
4717: // (rom) ReplaceStrings1, GetSubStr removed
4718: function GetLongFileName(const FileName: string): string;
4719: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4720: function GetParameter: string;
4721: function GetComputerID: string;
4722: function GetComputerName: string;
4723: {**** string routines }
4724: { ReplaceAllStrings searches for all substrings, Words,
4725:   in a string, S, and replaces them with Frases with the same Index. }
4726: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4727: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4728:   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 }
4729: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4730: { CountOfLines calculates the lines count in a string, S,
4731:   each line must be separated from another with CrLf sequence }
4732: function CountOfLines(const S: string): Integer;
4733: { DeleteLines deletes all lines from strings which in the words,  words.
4734:   The word of will be deleted from strings. }
4735: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4736: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4737:   Lines contained only spaces also deletes. }
4738: procedure DeleteEmptyLines(Ss: TStrings);
4739: { SQLAddWhere addes or modifies existing where-statement, where,
4740:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4741:   it must be started on the begining of any line }
4742: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4743: {**** files routines - }
4744: {$IFDEF MSWINDOWS}
4745: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4746:   Resource can be compressed using MS Compress program}
4747: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4748: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
     Boolean;
4749: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4750: {$ENDIF MSWINDOWS}
4751: { IniReadSection read section, Section, from ini-file,
4752:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4753:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4754: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4755: { LoadTextFile load text file, FileName, into string }
4756: function LoadTextFile(const FileName: TFileName): string;
4757: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4758: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4759: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4760: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4761: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4762: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4763: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4764: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4765: { RATextCalcHeight calculate needed height for
4766:   correct output, using RATextOut or RATextOutEx functions }
4767: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4768: { Cinema draws some visual effect }
4769: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4770: { Roughed fills rect with special 3D pattern }
4771: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4772: { 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 }
4773: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4774: { TextWidth calculate text with for writing using standard desktop font }
4775: function TextWidth(const AStr: string): Integer;
4776: { TextHeight calculate text height for writing using standard desktop font }
4777: function TextHeight(const AStr: string): Integer;
4778: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4779: procedure Error(const Msg: string);
4780: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4781:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4782: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4783: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4784:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4785: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4786:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4787: function ItemHtPlain(const Text: string): string;
4788: { ClearList - clears list of TObject }
4789: procedure ClearList(List: TList);
4790: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
```

```
4791: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4792: { RTTI support }
4793: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4794: function GetPropStr(Obj: TObject; const PropName: string): string;
4795: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4796: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4797: procedure PrepareIniSection(Ss: TStrings);
4798: { following functions are not documented because they are don't work properly, so don't use them }
4799: // (rom) from JvBandWindows to make it obsolete
4800: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4801: // (rom) from JvBandUtils to make it obsolete
4802: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4803: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4804: function CreateIconFromClipboard: TIcon;
4805: { begin JvIconClipboardUtils } { Icon clipboard routines }
4806: function CF_ICON: Word;
4807: procedure AssignClipboardIcon(Icon: TIcon);
4808: { Real-size icons support routines (32-bit only) }
4809: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4810: function CreateRealSizeIcon(Icon: TIcon): HICON;
4811: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4812: {end JvIconClipboardUtils }
4813: function CreateScreenCompatibleDC: HDC;
4814: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF}
4815: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4816: { begin JvRLE } // (rom) changed API for inclusion in JCL
4817: procedure RleCompressTo(InStream, OutStream: TStream);
4818: procedure RleDecompressTo(InStream, OutStream: TStream);
4819: procedure RleCompress(Stream: TStream);
4820: procedure RleDecompress(Stream: TStream);
4821: { end JvRLE } { begin JvDateUtil }
4822: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4823: function IsLeapYear(AYear: Integer): Boolean;
4824: function DaysInAMonth(const AYear, AMonth: Word): Word;
4825: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4826: function FirstDayOfPrevMonth: TDateTime;
4827: function LastDayOfPrevMonth: TDateTime;
4828: function FirstDayOfNextMonth: TDateTime;
4829: function ExtractDay(ADate: TDateTime): Word;
4830: function ExtractMonth(ADate: TDateTime): Word;
4831: function ExtractYear(ADate: TDateTime): Word;
4832: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4833: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4834: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4835: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4836: function ValidDate(ADate: TDateTime): Boolean;
4837: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4838: function MonthsBetween(Date1, Date2: TDateTime): Double;
4839: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4840: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4841: function DaysBetween(Date1, Date2: TDateTime): Longint;
4842: { The same as previous but if Date2 < Date1 result = 0 }
4843: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4844: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4845: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4846: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4847: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4848: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4849: { String to date conversions }
4850: function GetDateOrder(const DateFormat: string): TDateOrder;
4851: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4852: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4853: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4854: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4855: //function DefDateFormat(AFourDigitYear: Boolean): string;
4856: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4857: function FormatLongDate(Value: TDateTime): string;
4858: function FormatLongDateTime(Value: TDateTime): string;
4859: { end JvDateUtil }
4860: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4861: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4862: { begin JvStrUtils } { ** Common string handling routines ** }
4863: {$IFDEF UNIX}
4864: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4865:   const ToCode, FromCode: AnsiString): Boolean;
4866: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4867: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4868: function OemStrToAnsi(const S: AnsiString): AnsiString;
4869: function AnsiStrToOem(const S: AnsiString): AnsiString;
4870: {$ENDIF UNIX}
4871: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4872: { StrToOem translates a string from the Windows character set into the OEM character set. }
4873: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4874: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4875: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4876: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4877: function ReplaceStr(const S, Srch, Replace: string): string;
4878: { Returns string with every occurrence of Srch string replaced with Replace string. }
```

```
4879: function DelSpace(const S: string): string;
4880: { DelSpace return a string with all white spaces removed. }
4881: function DelChars(const S: string; Chr: Char): string;
4882: { DelChars return a string with all Chr characters removed. }
4883: function DelBSpace(const S: string): string;
4884: { DelBSpace trims leading spaces from the given string. }
4885: function DelESpace(const S: string): string;
4886: { DelESpace trims trailing spaces from the given string. }
4887: function DelRSpace(const S: string): string;
4888: { DelRSpace trims leading and trailing spaces from the given string. }
4889: function DelSpace1(const S: string): string;
4890: { DelSpace1 return a string with all non-single white spaces removed. }
4891: function Tab2Space(const S: string; Numb: Byte): string;
4892: { Tab2Space converts any tabulation character in the given string to the
4893:   Numb spaces characters. }
4894: function NPos(const C: string; S: string; N: Integer): Integer;
4895: { NPos searches for a N-th position of substring C in a given string. }
4896: function MakeStr(C: Char; N: Integer): string; overload;
4897: {$IFNDEF COMPILER12_UP}
4898: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4899: {$ENDIF !COMPILER12_UP}
4900: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4901: { MakeStr return a string of length N filled with character C. }
4902: function AddChar(C: Char; const S: string; N: Integer): string;
4903: { AddChar return a string left-padded to length N with characters C. }
4904: function AddCharR(C: Char; const S: string; N: Integer): string;
4905: { AddCharR return a string right-padded to length N with characters C. }
4906: function LeftStr(const S: string; N: Integer): string;
4907: { LeftStr return a string right-padded to length N with blanks. }
4908: function RightStr(const S: string; N: Integer): string;
4909: { RightStr return a string left-padded to length N with blanks. }
4910: function CenterStr(const S: string; Len: Integer): string;
4911: { CenterStr centers the characters in the string based upon the Len specified. }
4912: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4913: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4914:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4915: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4916: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4917: function Copy2Symb(const S: string; Symb: Char): string;
4918: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4919: function Copy2SymbDel(var S: string; Symb: Char): string;
4920: { Copy2SymbDel returns a substring of a string S from begining to first
4921:   character Symb and removes this substring from S. }
4922: function Copy2Space(const S: string): string;
4923: { Copy2Symb returns a substring of a string S from begining to first white space. }
4924: function Copy2SpaceDel(var S: string): string;
4925: { Copy2SpaceDel returns a substring of a string S from begining to first
4926:   white space and removes this substring from S. }
4927: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4928: { Returns string, with the first letter of each word in uppercase,
4929:   all other letters in lowercase. Words are delimited by WordDelims. }
4930: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4931: { WordCount given a set of word delimiters, returns number of words in S. }
4932: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4933: { Given a set of word delimiters, returns start position of N'th word in S. }
4934: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4935: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4936: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4937: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4938:   delimiters, return the N'th word in S. }
4939: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4940: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4941:   that started from position Pos. }
4942: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4943: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4944: function QuotedString(const S: string; Quote: Char): string;
4945: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4946: function ExtractQuotedString(const S: string; Quote: Char): string;
4947: { ExtractQuotedString removes the Quote characters from the beginning and
4948:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4949: function FindPart(const HelpWilds, InputStr: string): Integer;
4950: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4951: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4952: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4953: function XorString(const Key, Src: ShortString): ShortString;
4954: function XorEncode(const Key, Source: string): string;
4955: function XorDecode(const Key, Source: string): string;
4956: { ** Command line routines ** }
4957: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4958: { ** Numeric string handling routines ** }
4959: function Numb2USA(const S: string): string;
4960: { Numb2USA converts numeric string S to USA-format. }
4961: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4962: { Dec2Hex converts the given value to a hexadecimal string representation
4963:   with the minimum number of digits (A) specified. }
4964: function Hex2Dec(const S: string): Longint;
4965: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4966: function Dec2Numb(N: Int64; A, B: Byte): string;
4967: { Dec2Numb converts the given value to a string representation with the
```

```
4968:   base equal to B and with the minimum number of digits (A) specified. }
4969: function Numb2Dec(S: string; B: Byte): Int64;
4970: { Numb2Dec converts the given B-based numeric string to the corresponding
4971:   integer value. }
4972: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4973: { IntToBin converts the given value to a binary string representation
4974:   with the minimum number of digits specified. }
4975: function IntToRoman(Value: Longint): string;
4976: { IntToRoman converts the given value to a roman numeric string representation. }
4977: function RomanToInt(const S: string): Longint;
4978: { RomanToInt converts the given string to an integer value. If the string
4979:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4980: function FindNotBlankCharPos(const S: string): Integer;
4981: function FindNotBlankCharPosW(const S: WideString): Integer;
4982: function AnsiChangeCase(const S: string): string;
4983: function WideChangeCase(const S: string): string;
4984: function StartsText(const SubStr, S: string): Boolean;
4985: function EndsText(const SubStr, S: string): Boolean;
4986: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4987: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4988: {end JvStrUtils}
4989: {$IFDEF UNIX}
4990: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4991: {$ENDIF UNIX}
4992: { begin JvFileUtil }
4993: function FileDateTime(const FileName: string): TDateTime;
4994: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4995: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4996: function NormalDir(const DirName: string): string;
4997: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4998: function ValidFileName(const FileName: string): Boolean;
4999: {$IFDEF MSWINDOWS}
5000: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5001: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5002: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
5003: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
5004: {$ENDIF MSWINDOWS}
5005: function GetWindowsDir: string;
5006: function GetSystemDir: string;
5007: function ShortToLongFileName(const ShortName: string): string;
5008: function LongToShortFileName(const LongName: string): string;
5009: function ShortToLongPath(const ShortName: string): string;
5010: function LongToShortPath(const LongName: string): string;
5011: {$IFDEF MSWINDOWS}
5012: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
5013: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
5014: {$ENDIF MSWINDOWS}
5015: { end JvFileUtil }
5016: // Works like PtInRect but includes all edges in comparision
5017: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
5018: // Works like PtInRect but excludes all edges from comparision
5019: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
5020: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5021: function IsFourDigitYear: Boolean;
5022: { moved from JvJVCLUTils }
5023: //Open an object with the shell (url or something like that)
5024: function OpenObject(const Value: string): Boolean; overload;
5025: function OpenObject(Value: PChar): Boolean; overload;
5026: {$IFDEF MSWINDOWS}
5027: //Raise the last Exception
5028: procedure RaiseLastWin32; overload;
5029: procedure RaiseLastWin32(const Text: string); overload;
5030: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
5031: function GetFileVersion(const AFileName: string): Cardinal;
5032: {$EXTERNALSYM GetFileVersion}
5033: //Get version of Shell.dll
5034: function GetShellVersion: Cardinal;
5035: {$EXTERNALSYM GetShellVersion}
5036: // CD functions  on HW
5037: procedure OpenCdDrive;
5038: procedure CloseCdDrive;
5039: // returns True if Drive is accessible
5040: function DiskInDrive(Drive: Char): Boolean;
5041: {$ENDIF MSWINDOWS}
5042: //Same as linux function ;)
5043: procedure PError(const Text: string);
5044: // execute a program without waiting
5045: procedure Exec(const FileName, Parameters, Directory: string);
5046: // execute a program and wait for it to finish
5047: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5048: // returns True if this is the first instance of the program that is running
5049: function FirstInstance(const ATitle: string): Boolean;
5050: // restores a window based on it's classname and Caption. Either can be left empty
5051: // to widen the search
5052: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5053: // manipulate the traybar and start button
5054: procedure HideTraybar;
```

```
5055: procedure ShowTraybar;
5056: procedure ShowStartButton(Visible: Boolean = True);
5057: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5058: procedure MonitorOn;
5059: procedure MonitorOff;
5060: procedure LowPower;
5061: // send a key to the window named AppName
5062: function SendKey(const AppName: string; Key: Char): Boolean;
5063: {$IFDEF MSWINDOWS}
5064: // returns a list of all win currently visible, the Objects property is filled with their window handle
5065: procedure GetVisibleWindows(List: TStrings);
5066: Function GetVisibleWindowsF( List : TStrings):TStrings');
5067: // associates an extension to a specific program
5068: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5069: procedure AddToRecentDocs(const FileName: string);
5070: function GetRecentDocs: TStringList;
5071: {$ENDIF MSWINDOWS}
5072: function CharIsMoney(const Ch: Char): Boolean;
5073: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5074: function IntToExtended(I: Integer): Extended;
5075: { GetChangedText works out the new text given the current cursor pos & the key pressed
5076:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5077: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5078: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5079: //function StrIsInteger(const S: string): Boolean;
5080: function StrIsFloatMoney(const Ps: string): Boolean;
5081: function StrIsDateTime(const Ps: string): Boolean;
5082: function PreformatDateString(Ps: string): string;
5083: function BooleanToInteger(const B: Boolean): Integer;
5084: function StringToBoolean(const Ps: string): Boolean;
5085: function SafeStrToDateTime(const Ps: string): TDateTime;
5086: function SafeStrToDate(const Ps: string): TDateTime;
5087: function SafeStrToTime(const Ps: string): TDateTime;
5088: function StrDelete(const psSub, psMain: string): string;
5089:   { returns the fractional value of pcValue}
5090: function TimeOnly(pcValue: TDateTime): TTime;
5091: { returns the integral value of pcValue }
5092: function DateOnly(pcValue: TDateTime): TDate;
5093: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5094: const { TDateTime value used to signify Null value}
5095:   NullEquivalentDate: TDateTime = 0.0;
5096: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5097: // Replacement for Win32Check to avoid platform specific warnings in D6
5098: function OSCheck(RetVal: Boolean): Boolean;
5099: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5100:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5101:   not be forced to use FileCtrl unnecessarily }
5102: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5103: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5104: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5105: function MinimizeString(const S: string; const MaxLen: Integer): string;
5106: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
     SW_SHOWDEFAULT);
5107: { 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.}
5108: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5109: {$ENDIF MSWINDOWS}
5110: procedure ResourceNotFound(ResID: PChar);
5111: function EmptyRect: TRect;
5112: function RectWidth(R: TRect): Integer;
5113: function RectHeight(R: TRect): Integer;
5114: function CompareRect(const R1, R2: TRect): Boolean;
5115: procedure RectNormalize(var R: TRect);
5116: function RectIsSquare(const R: TRect): Boolean;
5117: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5118: //If AMaxSize = -1 ,then auto calc Square's max size
5119: {$IFDEF MSWINDOWS}
5120: procedure FreeUnusedOle;
5121: function GetWindowsVersion: string;
5122: function LoadDLL(const LibName: string): THandle;
5123: function RegisterServer(const ModuleName: string): Boolean;
5124: function UnregisterServer(const ModuleName: string): Boolean;
5125: {$ENDIF MSWINDOWS}
5126: { String routines }
5127: function GetEnvVar(const VarName: string): string;
5128: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5129: function StringToPChar(var S: string): PChar;
5130: function StrPAlloc(const S: string): PChar;
5131: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5132: function DropT(const S: string): string;
5133: { Memory routines }
5134: function AllocMemo(Size: Longint): Pointer;
5135: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5136: procedure FreeMemo(var fpBlock: Pointer);
5137: function GetMemoSize(fpBlock: Pointer): Longint;
5138: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5139: { Manipulate huge pointers routines }
5140: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
```

```
5141: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5142: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5143: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5144: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5145: function WindowClassName(Wnd: THandle): string;
5146: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5147: procedure ActivateWindow(Wnd: THandle);
5148: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5149: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5150: { SetWindowTop put window to top without recreating window }
5151: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5152: procedure CenterWindow(Wnd: THandle);
5153: function MakeVariant(const Values: array of Variant): Variant;
5154: { Convert dialog units to pixels and backwards }
5155: {$IFDEF MSWINDOWS}
5156: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5157: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5158: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5159: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5160: {$ENDIF MSWINDOWS}
5161: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5162: {$IFDEF BCB}
5163: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5164: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5165: {$ELSE}
5166: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5167: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5168: {$ENDIF BCB}
5169: {$IFDEF MSWINDOWS}
5170: { BrowseForFolderNative displays Browse For Folder dialog }
5171: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5172: {$ENDIF MSWINDOWS}
5173: procedure AntiAlias(Clip: TBitmap);
5174: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5175: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5176:   ABitmap: TBitmap; const SourceRect: TRect);
5177: function IsTrueType(const FontName: string): Boolean;
5178: // Removes all non-numeric characters from AValue and returns the resulting string
5179: function TextToValText(const AValue: string): string;
5180: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5181: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5182: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5183: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5184: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5185:
5186: ******************************************************unit uPSI_JvTFUtils;
5187:  Function JExtractYear( ADate : TDateTime) : Word
5188:  Function JExtractMonth( ADate : TDateTime) : Word
5189:  Function JExtractDay( ADate : TDateTime) : Word
5190:  Function ExtractHours( ATime : TDateTime) : Word
5191:  Function ExtractMins( ATime : TDateTime) : Word
5192:  Function ExtractSecs( ATime : TDateTime) : Word
5193:  Function ExtractMSecs( ATime : TDateTime) : Word
5194:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5195:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5196:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5197:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5198:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5199:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5200:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5201:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5202:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5203:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5204:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5205:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5206:  Procedure EnsureMonth( Month : Word)
5207:  Procedure EnsureDOW( DOW : Word)
5208:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5209:  Function Lesser( N1, N2 : Integer) : Integer
5210:  Function Greater( N1, N2 : Integer) : Integer
5211:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5212:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5213:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5214:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5215:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5216:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5217:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
     AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5218:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
     HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5219:  Function JRectWidth( ARect : TRect) : Integer
5220:  Function JRectHeight( ARect : TRect) : Integer
5221:  Function JEmptyRect : TRect
5222:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5223:
5224: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5225: begin
5226:  Procedure HideTaskBarButton( hWindow : HWND)
```

```
5227:  Function msLoadStr( ID : Integer) : String
5228:  Function msFormat( fmt : String; params : array of const) : String
5229:  Function msFileExists( const FileName : String) : Boolean
5230:  Function msIntToStr( Int : Int64) : String
5231:  Function msStrPas( const Str : PChar) : String
5232:  Function msRenameFile( const OldName, NewName : String) : Boolean
5233:  Function CutFileName( s : String) : String
5234:  Function GetVersionInfo( var VersionString : String) : DWORD
5235:  Function FormatTime( t : Cardinal) : String
5236:  Function msCreateDir( const Dir : string) : Boolean
5237:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5238:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5239:  Function msStrLen( Str : PChar) : Integer
5240:  Function msDirectoryExists( const Directory : String) : Boolean
5241:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5242:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5243:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5244:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5245:  Function GetTextFromFile( Filename : String) : string
5246:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5247:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5248:  Function msStrToInt( s : String) : Integer
5249:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
5250: end;
5251:
5252: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5253: begin
5254:   //TDynFloatArray', 'array of Extended
5255:   TDynLWordArray', 'array of LongWord
5256:   TDynLIntArray', 'array of LongInt
5257:   TDynFloatMatrix', 'array of TDynFloatArray
5258:   TDynLWordMatrix', 'array of TDynLWordArray
5259:   TDynLIntMatrix', 'array of TDynLIntArray
5260:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5261:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5262:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5263:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5264:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5265:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5266:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5267:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5268:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5269:  Function MNorm( const X : TDynFloatArray) : Extended
5270:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5271:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5272:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5273:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5274:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5275:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5276:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5277:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5278:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5279:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5280:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5281:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5282:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5283: end;
5284:
5285: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5286: begin
5287:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5288:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5289:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5290:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5291:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5292:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5293:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5294:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5295:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5296:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5297:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5298:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5299:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5300:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5301:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5302:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5303:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5304:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5305:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5306:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5307:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5308:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5309:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5310:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5311:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5312:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5313:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5314:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5315:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
```

```
5316:  'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5317:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5318:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5319:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5320:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5321:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5322:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5323:  'ESBPi','Extended').setExtended( 3.14159265358979323385);
5324:  'ESBInvPi','Extended').setExtended( 3.18309886183790671154e-1);
5325:  'ESBTwoPi','Extended').setExtended( 6.28318530717958647769);
5326:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5327:  'ESBPi2','Extended').setExtended( 9.86960440108935861188);
5328:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5329:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5330:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5331:  'ESBPiOn4','Extended').setExtended( 0.78539816339744830961);
5332:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5333:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5334:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5335:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5336:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5337:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5338:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5339:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5340:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5341:   //LongWord', 'Cardinal
5342:   TBitList', 'Word
5343:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5344:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5345:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5346:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5347:  Function SameFloat( const X1, X2 : Extended) : Boolean
5348:  Function FloatIsZero( const X : Extended) : Boolean
5349:  Function FloatIsPositive( const X : Extended) : Boolean
5350:  Function FloatIsNegative( const X : Extended) : Boolean
5351:  Procedure IncLim( var B : Byte; const Limit : Byte)
5352:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5353:  Procedure IncLimW( var B : Word; const Limit : Word)
5354:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5355:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5356:  Procedure DecLim( var B : Byte; const Limit : Byte)
5357:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5358:  Procedure DecLimW( var B : Word; const Limit : Word)
5359:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5360:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5361:  Function MaxB( const B1, B2 : Byte) : Byte
5362:  Function MinB( const B1, B2 : Byte) : Byte
5363:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5364:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5365:  Function MaxW( const B1, B2 : Word) : Word
5366:  Function MinW( const B1, B2 : Word) : Word
5367:  Function esbMaxI( const B1, B2 : Integer) : Integer
5368:  Function esbMinI( const B1, B2 : Integer) : Integer
5369:  Function MaxL( const B1, B2 : LongInt) : LongInt
5370:  Function MinL( const B1, B2 : LongInt) : LongInt
5371:  Procedure SwapB( var B1, B2 : Byte)
5372:  Procedure SwapSI( var B1, B2 : ShortInt)
5373:  Procedure SwapW( var B1, B2 : Word)
5374:  Procedure SwapI( var B1, B2 : SmallInt)
5375:  Procedure SwapL( var B1, B2 : LongInt)
5376:  Procedure SwapI32( var B1, B2 : Integer)
5377:  Procedure SwapC( var B1, B2 : LongWord)
5378:  Procedure SwapInt64( var X, Y : Int64)
5379:  Function esbSign( const B : LongInt) : ShortInt
5380:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5381:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5382:  Function Max3Word( const X1, X2, X3 : Word) : Word
5383:  Function Min3Word( const X1, X2, X3 : Word) : Word
5384:  Function MaxBArray( const B : array of Byte) : Byte
5385:  Function MaxWArray( const B : array of Word) : Word
5386:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5387:  Function MaxIArray( const B : array of Integer) : Integer
5388:  Function MaxLArray( const B : array of LongInt) : LongInt
5389:  Function MinBArray( const B : array of Byte) : Byte
5390:  Function MinWArray( const B : array of Word) : Word
5391:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5392:  Function MinIArray( const B : array of Integer) : Integer
5393:  Function MinLArray( const B : array of LongInt) : LongInt
5394:  Function SumBArray( const B : array of Byte) : Byte
5395:  Function SumBArray2( const B : array of Byte) : Word
5396:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5397:  Function SumSIArray2( const B : array of ShortInt) : Integer
5398:  Function SumWArray( const B : array of Word) : Word
5399:  Function SumWArray2( const B : array of Word) : LongInt
5400:  Function SumIArray( const B : array of Integer) : Integer
5401:  Function SumLArray( const B : array of LongInt) : LongInt
5402:  Function SumLWArray( const B : array of LongWord) : LongWord
5403:  Function ESBDigits( const X : LongWord) : Byte
5404:  Function BitsHighest( const X : LongWord) : Integer
```

```
5405:  Function ESBBitsNeeded( const X : LongWord) : Integer
5406:  Function esbGCD( const X, Y : LongWord) : LongWord
5407:  Function esbLCM( const X, Y : LongInt) : Int64
5408:  //Function esbLCM( const X, Y : LongInt) : LongInt
5409:  Function RelativePrime( const X, Y : LongWord) : Boolean
5410:  Function Get87ControlWord : TBitList
5411:  Procedure Set87ControlWord( const CWord : TBitList)
5412:  Procedure SwapExt( var X, Y : Extended)
5413:  Procedure SwapDbl( var X, Y : Double)
5414:  Procedure SwapSing( var X, Y : Single)
5415:  Function esbSgn( const X : Extended) : ShortInt
5416:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5417:  Function ExtMod( const X, Y : Extended) : Extended
5418:  Function ExtRem( const X, Y : Extended) : Extended
5419:  Function CompMOD( const X, Y : Comp) : Comp
5420:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5421:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5422:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5423:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5424:  Function MaxExt( const X, Y : Extended) : Extended
5425:  Function MinExt( const X, Y : Extended) : Extended
5426:  Function MaxEArray( const B : array of Extended) : Extended
5427:  Function MinEArray( const B : array of Extended) : Extended
5428:  Function MaxSArray( const B : array of Single) : Single
5429:  Function MinSArray( const B : array of Single) : Single
5430:  Function MaxCArray( const B : array of Comp) : Comp
5431:  Function MinCArray( const B : array of Comp) : Comp
5432:  Function SumSArray( const B : array of Single) : Single
5433:  Function SumEArray( const B : array of Extended) : Extended
5434:  Function SumSqEArray( const B : array of Extended) : Extended
5435:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5436:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5437:  Function SumCArray( const B : array of Comp) : Comp
5438:  Function FactorialX( A : LongWord) : Extended
5439:  Function PermutationX( N, R : LongWord) : Extended
5440:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5441:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5442:  Function esbGeometricMean( const X : array of Extended) : Extended
5443:  Function esbHarmonicMean( const X : array of Extended) : Extended
5444:  Function ESBMean( const X : array of Extended) : Extended
5445:  Function esbSampleVariance( const X : array of Extended) : Extended
5446:  Function esbPopulationVariance( const X : array of Extended) : Extended
5447:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5448:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5449:  Function GetMedian( const SortedX : array of Extended) : Extended
5450:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5451:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5452:  Function ESBMagnitude( const X : Extended) : Integer
5453:  Function ESBTan( Angle : Extended) : Extended
5454:  Function ESBCot( Angle : Extended) : Extended
5455:  Function ESBCosec( const Angle : Extended) : Extended
5456:  Function ESBSec( const Angle : Extended) : Extended
5457:  Function ESBArcTan( X, Y : Extended) : Extended
5458:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5459:  Function ESBArcCos( const X : Extended) : Extended
5460:  Function ESBArcSin( const X : Extended) : Extended
5461:  Function ESBArcSec( const X : Extended) : Extended
5462:  Function ESBArcCosec( const X : Extended) : Extended
5463:  Function ESBLog10( const X : Extended) : Extended
5464:  Function ESBLog2( const X : Extended) : Extended
5465:  Function ESBLogBase( const X, Base : Extended) : Extended
5466:  Function Pow2( const X : Extended) : Extended
5467:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5468:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5469:  Function XtoY( const X, Y : Extended) : Extended
5470:  Function esbTenToY( const Y : Extended) : Extended
5471:  Function esbTwoToY( const Y : Extended) : Extended
5472:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5473:  Function esbISqrt( const I : LongWord) : Longword
5474:  Function ILog2( const I : LongWord) : LongWord
5475:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5476:  Function ESBArCosh( X : Extended) : Extended
5477:  Function ESBArSinh( X : Extended) : Extended
5478:  Function ESBArTanh( X : Extended) : Extended
5479:  Function ESBCosh( X : Extended) : Extended
5480:  Function ESBSinh( X : Extended) : Extended
5481:  Function ESBTanh( X : Extended) : Extended
5482:  Function InverseGamma( const X : Extended) : Extended
5483:  Function esbGamma( const X : Extended) : Extended
5484:  Function esbLnGamma( const X : Extended) : Extended
5485:  Function esbBeta( const X, Y : Extended) : Extended
5486:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5487: end;
5488:
5489: ********************************Integer Huge Cardinal Utils**************************
5490: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5491: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5492: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5493: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
```

```
5494: Function BitCount_8( Value : byte) : integer
5495: Function BitCount_16( Value : uint16) : integer
5496: Function BitCount_32( Value : uint32) : integer
5497: Function BitCount_64( Value : uint64) : integer
5498: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5499:  Procedure ( CountPrimalityTests : integer)
5500:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5501:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5502:  Function isCoPrime( a, b : THugeCardinal) : boolean
5503:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5504:  Function hasSmallFactor( p : THugeCardinal) : boolean
5505:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer) : boolean
5506:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5507:  Const('StandardExponent','LongInt'( 65537);
5508: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5509:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5510:
5511: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5512: begin
5513:   AddTypeS('TXRTLInteger', 'array of Integer
5514:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5515:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5516:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5517:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5518:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5519:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5520:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5521:   'BitsPerByte','LongInt'( 8);
5522:   BitsPerDigit','LongInt'( 32);
5523:   SignBitMask','LongWord( $80000000);
5524:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5525:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5526:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5527:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5528:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5529:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5530:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5531:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5532:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5533:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5534:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5535:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5536:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5537:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5538:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5539:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5540:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5541:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5542:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5543:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5544:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5545:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5546:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5547:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5548:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5549:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5550:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5551:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5552:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5553:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5554:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5555:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5556:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5557:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5558:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5559:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5560:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5561:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger)
5562:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5563:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5564:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5565:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5566:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5567:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5568:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5569:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5570:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5571:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5572:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5573:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5574:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5575:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5576:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
```

```
5577:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5578:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5579:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5580:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5581:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5582:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5583:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5584:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5585:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5586:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5587:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5588:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5589:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5590:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5591:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5592:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5593:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5594:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5595:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5596:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5597:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5598: end;
5599:
5600: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5601: begin
5602:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5603:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5604:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5605:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5606:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5607:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5608:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5609:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5610:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5611:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5612:  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)
5613: end;
5614:
5615:
5616: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5617: begin
5618:  Function StrDec( S : String) : String
5619:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5620:  Function ReadNumFromEdit( Edit : TEdit) : Float
5621:  Procedure WriteNumToFile( var F : Text; X : Float)
5622: end;
5623:
5624: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5625: begin
5626:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5627:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5628:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5629:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5630:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5631:  Procedure TeX_SetGraphTitle( Title : String)
5632:  Procedure TeX_SetOxTitle( Title : String)
5633:  Procedure TeX_SetOyTitle( Title : String)
5634:  Procedure TeX_PlotOxAxis
5635:  Procedure TeX_PlotOyAxis
5636:  Procedure TeX_PlotGrid( Grid : TGrid)
5637:  Procedure TeX_WriteGraphTitle
5638:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5639:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5640:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5641:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5642:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5643:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5644:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5645:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5646:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5647:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5648:  Function Xcm( X : Float) : Float
5649:  Function Ycm( Y : Float) : Float
5650: end;
5651:
5652: *---------------------------------------------------------------------------*)
5653: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5654: begin
5655:   TConstArray', 'array of TVarRec
5656:  Function CopyVarRec( const Item : TVarRec) : TVarRec
5657:  Function CreateConstArray( const Elements : array of const) : TConstArray
5658:  Procedure FinalizeVarRec( var Item : TVarRec)
```

```
5659:  Procedure FinalizeConstArray( var Arr : TConstArray)
5660: end;
5661:
5662: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5663: begin
5664:  Function HexBS( B : Byte) : ShortString
5665:  Function HexWS( W : Word) : ShortString
5666:  Function HexLS( L : LongInt) : ShortString
5667:  Function HexPtrS( P : Pointer) : ShortString
5668:  Function BinaryBS( B : Byte) : ShortString
5669:  Function BinaryWS( W : Word) : ShortString
5670:  Function BinaryLS( L : LongInt) : ShortString
5671:  Function OctalBS( B : Byte) : ShortString
5672:  Function OctalWS( W : Word) : ShortString
5673:  Function OctalLS( L : LongInt) : ShortString
5674:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5675:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5676:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5677:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5678:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5679:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5680:  Function Long2StrS( L : LongInt) : ShortString
5681:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5682:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5683:  Function ValPrepS( const S : ShortString) : ShortString
5684:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5685:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5686:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5687:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5688:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5689:  Function TrimLeadS( const S : ShortString) : ShortString
5690:  Function TrimTrailS( const S : ShortString) : ShortString
5691:  Function TrimS( const S : ShortString) : ShortString
5692:  Function TrimSpacesS( const S : ShortString) : ShortString
5693:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5694:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5695:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5696:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5697:  Function ScrambleS( const S, Key : ShortString) : ShortString
5698:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5699:  Function FilterS( const S, Filters : ShortString) : ShortString
5700:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5701:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5702:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5703:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5704:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5705:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5706:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5707:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5708:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
5709:  Function CompStringS( const S1, S2 : ShortString) : Integer
5710:  Function CompUCStringS( const S1, S2 : ShortString) : Integer
5711:  Function SoundexS( const S : ShortString) : ShortString
5712:  Function MakeLetterSetS( const S : ShortString) : Longint
5713:  Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5714:  Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5715:  Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
      Pos:Cardinal):Bool;
5716:  Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5717:  Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5718:  Function JustFilenameS( const PathName : ShortString) : ShortString
5719:  Function JustNameS( const PathName : ShortString) : ShortString
5720:  Function JustExtensionS( const Name : ShortString) : ShortString
5721:  Function JustPathnameS( const PathName : ShortString) : ShortString
5722:  Function AddBackSlashS( const DirName : ShortString) : ShortString
5723:  Function CleanPathNameS( const PathName : ShortString) : ShortString
5724:  Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5725:  Function CommaizeS( L : LongInt) : ShortString
5726:  Function CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5727:  Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
      DecPt:Char):ShortString;
5728:  Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5729:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5730:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5731:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5732:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5733:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5734:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5735:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5736:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5737:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5738:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5739:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5740:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5741:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5742:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
      SubString:ShortString):Bool;
```

```
5743:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
       SubStr:ShortString):Bool;
5744:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
       SubString:ShortString):Bool;
5745:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
       SubString:ShortString):Bool;
5746:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5747:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5748:  Function ExtractTokensS(const S,
       Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5749:  Function IsChAlphaS( C : Char) : Boolean
5750:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5751:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5752:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5753:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5754:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5755:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5756:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5757:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5758:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5759:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5760:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
       Cardinal):ShortString;
5761:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
       Replacements:Cardinal):ShortString;
5762:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5763:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
       Replacements:Cardinal):ShortString
5764:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
       Replacements:Cardinal):ShortString
5765:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5766:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5767:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5768:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5769: end;
5770:
5771:
5772: ********unit uPSI_StUtils; from Systools4*************************************************************
5773: Function SignL( L : LongInt) : Integer
5774: Function SignF( F : Extended) : Integer
5775: Function MinWord( A, B : Word) : Word
5776: Function MidWord( W1, W2, W3 : Word) : Word
5777: Function MaxWord( A, B : Word) : Word
5778: Function MinLong( A, B : LongInt) : LongInt
5779: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5780: Function MaxLong( A, B : LongInt) : LongInt
5781: Function MinFloat( F1, F2 : Extended) : Extended
5782: Function MidFloat( F1, F2, F3 : Extended) : Extended
5783: Function MaxFloat( F1, F2 : Extended) : Extended
5784: Function MakeInteger16( H, L : Byte) : SmallInt
5785: Function MakeWordS( H, L : Byte) : Word
5786: Function SwapNibble( B : Byte) : Byte
5787: Function SwapWord( L : LongInt) : LongInt
5788: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5789: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5790: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5791: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5792: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5793: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5794: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5795: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5796: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5797: Procedure ExchangeBytes( var I, J : Byte)
5798: Procedure ExchangeWords( var I, J : Word)
5799: Procedure ExchangeLongInts( var I, J : LongInt)
5800: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5801: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5802: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5803: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5804: //******************uPSI_StFIN;*************************************************************
5805: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5806: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
       Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5807: Function BondDuration( Settlement,Maturity:TStDate;Rate,
       Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5808: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
       Extended
5809: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
       Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5810: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
       Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5811: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5812: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5813: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5814: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5815: Function DollarToDecimalText( DecDollar : Extended) : string
5816: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5817: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5818: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
```

```
5819: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5820: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5821: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5822: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5823: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5824: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5825: Function IsCardValid( const S : string) : Boolean
5826: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5827: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5828: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5829: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5830: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5831: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5832: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5833: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5834: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5835: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5836: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5837: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5838: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5839: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5840: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5841: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5842: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5843: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5844: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5845: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5846:
5847: //*****************************************unit uPSI_StAstroP;
5848: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5849: //*****unit unit uPSI_StStat; Statistic Package of SysTools***************************************
5850: Function AveDev( const Data : array of Double) : Double
5851: Function AveDev16( const Data, NData : Integer) : Double
5852: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5853: Function Correlation( const Data1, Data2 : array of Double) : Double
5854: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5855: Function Covariance( const Data1, Data2 : array of Double) : Double
5856: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5857: Function DevSq( const Data : array of Double) : Double
5858: Function DevSq16( const Data, NData : Integer) : Double
5859: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5860:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5861: Function GeometricMeanS( const Data : array of Double) : Double
5862: Function GeometricMean16( const Data, NData : Integer) : Double
5863: Function HarmonicMeanS( const Data : array of Double) : Double
5864: Function HarmonicMean16( const Data, NData : Integer) : Double
5865: Function Largest( const Data : array of Double; K : Integer) : Double
5866: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5867: Function MedianS( const Data : array of Double) : Double
5868: Function Median16( const Data, NData : Integer) : Double
5869: Function Mode( const Data : array of Double) : Double
5870: Function Mode16( const Data, NData : Integer) : Double
5871: Function Percentile( const Data : array of Double; K : Double) : Double
5872: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5873: Function PercentRank( const Data : array of Double; X : Double) : Double
5874: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5875: Function Permutations( Number, NumberChosen : Integer) : Extended
5876: Function Combinations( Number, NumberChosen : Integer) : Extended
5877: Function FactorialS( N : Integer) : Extended
5878: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5879: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5880: Function Smallest( const Data : array of Double; K : Integer) : Double
5881: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5882: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5883: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5884:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5885:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5886: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5887: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5888: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5889: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5890: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5891: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5892: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5893: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5894: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5895: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5896: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5897: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5898: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
```

```
5899: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5900: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5901: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5902: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5903: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5904: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5905: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5906: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5907: Function NormSDist( Z : Single) : Single
5908: Function NormSInv( Probability : Single) : Single
5909: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5910: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5911: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5912: Function Erfc( X : Single) : Single
5913: Function GammaLn( X : Single) : Single
5914: Function LargestSort( const Data : array of Double; K : Integer) : Double
5915: Function SmallestSort( const Data : array of double; K : Integer) : Double
5916:
5917: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5918:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5919:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5920:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5921:  Function DefaultMergeName( MergeNum : Integer) : string
5922:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5923:
5924: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5925: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5926:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5927:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5928:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5929:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5930:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5931:  Function LunarPhase( UT : TStDateTimeRec) : Double
5932:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5933:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5934:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5935:  Function FullMoon( D : TStDate) : TStLunarRecord
5936:  Function LastQuarter( D : TStDate) : TStLunarRecord
5937:  Function NewMoon( D : TStDate) : TStLunarRecord
5938:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5939:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5940:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5941:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5942:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5943:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5944:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5945:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5946:  Function SiderealTime( UT : TStDateTimeRec) : Double
5947:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5948:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5949:  Function SEaster( Y, Epoch : Integer) : TStDate
5950:  Function DateTimeToAJD( D : TDateTime) : Double
5951:  Function HoursMin( RA : Double) : ShortString
5952:  Function DegsMin( DC : Double) : ShortString
5953:  Function AJDToDateTime( D : Double) : TDateTime
5954:
5955: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5956:  Function CurrentDate : TStDate
5957:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5958:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5959:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5960:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5961:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5962:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5963:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5964:  Function WeekOfYear( Julian : TStDate) : Byte
5965:  Function AstJulianDate( Julian : TStDate) : Double
5966:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5967:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5968:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5969:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5970:  Function StIsLeapYear( Year : Integer) : Boolean
5971:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5972:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5973:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5974:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5975:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5976:  Function CurrentTime : TStTime
5977:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5978:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5979:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5980:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5981:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5982:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5983:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5984:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5985:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5986:  Function StDateToDateTime( D : TStDate) : TDateTime
5987:  Function StTimeToDateTime( T : TStTime) : TDateTime
```

```
5988:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5989:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5990:
5991: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5992: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5993:  Function MonthToString( const Month : Integer) : string
5994:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5995:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5996:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5997:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5998:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5999:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
6000:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
6001:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
6002:  Function TimeStringToStTime( const Picture, S : string) : TStTime
6003:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
6004:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
6005:  Function DateStringIsBlank( const Picture, S : string) : Boolean
6006:  Function InternationalDate( ForceCentury : Boolean) : string
6007:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
6008:  Function InternationalTime( ShowSeconds : Boolean) : string
6009:  Procedure ResetInternationalInfo
6010:
6011: procedure SIRegister_StBase(CL: TPSPascalCompiler);
6012:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
6013:  Function AnsiUpperCaseShort32( const S : string) : string
6014:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
6015:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
6016:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
6017:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
6018:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
6019:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
6020:  Function Upcase( C : AnsiChar) : AnsiChar
6021:  Function LoCase( C : AnsiChar) : AnsiChar
6022:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
6023:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
6024:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6025:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6026:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6027:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
6028:  Procedure RaiseContainerError( Code : longint)
6029:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
6030:  Function ProductOverflow( A, B : LongInt) : Boolean
6031:  Function StNewStr( S : string) : PShortString
6032:  Procedure StDisposeStr( PS : PShortString)
6033:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
6034:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
6035:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
6036:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6037:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6038:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
6039:
6040: procedure SIRegister_usvd(CL: TPSPascalCompiler);
6041: begin
6042:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
6043:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
6044:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
6045:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
6046:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6047: end;
6048:
6049: //**********unit unit ; StMath Package of SysTools*****************************************
6050: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6051: Function PowerS( Base, Exponent : Extended) : Extended
6052: Function StInvCos( X : Double) : Double
6053: Function StInvSin( Y : Double) : Double
6054: Function StInvTan2( X, Y : Double) : Double
6055: Function StTan( A : Double) : Double
6056: Procedure DumpException;  //unit StExpEng;
6057: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6058:
6059: //**********unit unit ; StCRC Package of SysTools*****************************************
6060: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6061: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6062: Function Adler32OfFile( FileName : AnsiString) : LongInt
6063: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6064: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6065: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6066: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6067: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6068: Function Crc32OfFile( FileName : AnsiString) : LongInt
6069: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6070: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6071: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6072: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6073: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6074: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6075:
6076: //**********unit unit ; StBCD Package of SysTools*****************************************
```

```
6077: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6078: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6079: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6080: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6081: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6082: Function NegBcd( const B : TbcdS) : TbcdS
6083: Function AbsBcd( const B : TbcdS) : TbcdS
6084: Function FracBcd( const B : TbcdS) : TbcdS
6085: Function IntBcd( const B : TbcdS) : TbcdS
6086: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6087: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6088: Function ValBcd( const S : string) : TbcdS
6089: Function LongBcd( L : LongInt) : TbcdS
6090: Function ExtBcd( E : Extended) : TbcdS
6091: Function ExpBcd( const B : TbcdS) : TbcdS
6092: Function LnBcd( const B : TbcdS) : TbcdS
6093: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6094: Function PowBcd( const B, E : TbcdS) : TbcdS
6095: Function SqrtBcd( const B : TbcdS) : TbcdS
6096: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6097: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6098: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6099: Function IsIntBcd( const B : TbcdS) : Boolean
6100: Function TruncBcd( const B : TbcdS) : LongInt
6101: Function BcdExt( const B : TbcdS) : Extended
6102: Function RoundBcd( const B : TbcdS) : LongInt
6103: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6104: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6105: Function FormatBcd( const Format : string; const B : TbcdS) : string
6106: Function StrGeneralBcd( const B : TbcdS) : string
6107: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6108: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6109:
6110: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6111: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6112: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6113: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6114: Function StDeEscape( const EscStr : AnsiString) : Char
6115: Function StDoEscape( Delim : Char) : AnsiString
6116: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6117: Function AnsiHashText( const S : string; Size : Integer) : Integer
6118: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6119: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6120: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6121:
6122: //**********unit unit ; StNetCon Package of SysTools*****************************************
6123:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6124:     Constructor Create( AOwner : TComponent)
6125:     Function Connect : DWord
6126:     Function Disconnect : DWord
6127:     RegisterProperty('Password', 'String', iptrw);
6128:     Property('UserName', 'String', iptrw);
6129:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6130:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6131:     Property('LocalDevice', 'String', iptrw);
6132:     Property('ServerName', 'String', iptrw);
6133:     Property('ShareName', 'String', iptrw);
6134:     Property('OnConnect', 'TNotifyEvent', iptrw);
6135:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6136:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6137:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6138:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6139:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6140:   end;
6141: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6142: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6143: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6144: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6145: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6146: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6147: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6148: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6149: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6150: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6151: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6152: Function SuspendThread( hThread : THandle) : DWORD
6153: Function ResumeThread( hThread : THandle) : DWORD
6154: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6155: Function GetCurrentThread : THandle
6156: Procedure ExitThread( dwExitCode : DWORD)
6157: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6158: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6159: Procedure EndThread(ExitCode: Integer);
6160: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6161: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6162: Procedure FreeProcInstance( Proc : FARPROC)
6163: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6164: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
```

```
6165: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6166: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6167: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6168: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6169: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6170: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6171: Function CurrentParallelJobInfo : TParallelJobInfo
6172: Function ObtainParallelJobInfo : TParallelJobInfo
6173: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6174: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6175: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6176: Function
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL');
6177: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL');
6178: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD) : BOOL');
6179: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6180: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6181:
6182: ************************************************unit uPSI_JclMime;
6183:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6184:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6185:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6186:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6187:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6188:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6189:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6190:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6191:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6192:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6193:
6194: ************************************************unit uPSI_JclPrint;
6195:  Procedure DirectPrint( const Printer, Data : string)
6196:  Procedure SetPrinterPixelsPerInch
6197:  Function GetPrinterResolution : TPoint
6198:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6199:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6200:
6201:
6202: //*************************************unit uPSI_ShLwApi;***********************************
6203:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6204:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6205:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6206:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6207:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6208:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6209:  Function StrDup( lpSrch : PChar) : PChar
6210:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6211:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6212:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6213:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6214: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6215:  Function StrPBrk( psz, pszSet : PChar) : PChar
6216:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6217:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6218:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6219:  Function StrSpn( psz, pszSet : PChar) : Integer
6220:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6221:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6222:  Function StrToInt( lpSrch : PChar) : Integer
6223:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6224:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6225:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6226:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6227:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6228:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6229:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6230:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6231:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6232:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6233:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6234:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6235:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6236:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6237:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6238:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6239:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6240:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6241:  STIF_DEFAULT','LongWord( $00000000);
6242:  STIF_SUPPORT_HEX','LongWord( $00000001);
6243:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6244:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6245:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6246:  Function PathAddBackslash( pszPath : PChar) : PChar
6247:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6248:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
```

```
6249:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6250:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6251:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6252:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6253:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6254:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6255:  Function PathFileExists( pszPath : PChar) : BOOL
6256:  Function PathFindExtension( pszPath : PChar) : PChar
6257:  Function PathFindFileName( pszPath : PChar) : PChar
6258:  Function PathFindNextComponent( pszPath : PChar) : PChar
6259:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6260:  Function PathGetArgs( pszPath : PChar) : PChar
6261:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6262:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6263:  Function PathGetCharType( ch : Char) : UINT
6264:  GCT_INVALID','LongWord( $0000);
6265:  GCT_LFNCHAR','LongWord( $0001);
6266:  GCT_SHORTCHAR','LongWord( $0002);
6267:  GCT_WILD','LongWord( $0004);
6268:  GCT_SEPARATOR','LongWord( $0008);
6269:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6270:  Function PathIsDirectory( pszPath : PChar) : BOOL
6271:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6272:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6273:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6274:  Function PathIsRelative( pszPath : PChar) : BOOL
6275:  Function PathIsRoot( pszPath : PChar) : BOOL
6276:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6277:  Function PathIsUNC( pszPath : PChar) : BOOL
6278:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6279:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6280:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6281:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6282:  Function PathIsURL( pszPath : PChar) : BOOL
6283:  Function PathMakePretty( pszPath : PChar) : BOOL
6284:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6285:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6286:  Procedure PathQuoteSpaces( lpsz : PChar)
6287:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6288:  Procedure PathRemoveArgs( pszPath : PChar)
6289:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6290:  Procedure PathRemoveBlanks( pszPath : PChar)
6291:  Procedure PathRemoveExtension( pszPath : PChar)
6292:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6293:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6294:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6295:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6296:  Function PathSkipRoot( pszPath : PChar) : PChar
6297:  Procedure PathStripPath( pszPath : PChar)
6298:  Function PathStripToRoot( pszPath : PChar) : BOOL
6299:  Procedure PathUnquoteSpaces( lpsz : PChar)
6300:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6301:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6302:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6303:  Procedure PathUndecorate( pszPath : PChar)
6304:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6305:  URL_SCHEME_INVALID','LongInt'( - 1);
6306:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6307:  URL_SCHEME_FTP','LongInt'( 1);
6308:  URL_SCHEME_HTTP','LongInt'( 2);
6309:  URL_SCHEME_GOPHER','LongInt'( 3);
6310:  URL_SCHEME_MAILTO','LongInt'( 4);
6311:  URL_SCHEME_NEWS','LongInt'( 5);
6312:  URL_SCHEME_NNTP','LongInt'( 6);
6313:  URL_SCHEME_TELNET','LongInt'( 7);
6314:  URL_SCHEME_WAIS','LongInt'( 8);
6315:  URL_SCHEME_FILE','LongInt'( 9);
6316:  URL_SCHEME_MK','LongInt'( 10);
6317:  URL_SCHEME_HTTPS','LongInt'( 11);
6318:  URL_SCHEME_SHELL','LongInt'( 12);
6319:  URL_SCHEME_SNEWS','LongInt'( 13);
6320:  URL_SCHEME_LOCAL','LongInt'( 14);
6321:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6322:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6323:  URL_SCHEME_ABOUT','LongInt'( 17);
6324:  URL_SCHEME_RES','LongInt'( 18);
6325:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6326:  URL_SCHEME', 'Integer
6327:  URL_PART_NONE','LongInt'( 0);
6328:  URL_PART_SCHEME','LongInt'( 1);
6329:  URL_PART_HOSTNAME','LongInt'( 2);
6330:  URL_PART_USERNAME','LongInt'( 3);
6331:  URL_PART_PASSWORD','LongInt'( 4);
6332:  URL_PART_PORT','LongInt'( 5);
6333:  URL_PART_QUERY','LongInt'( 6);
6334:  URL_PART', 'DWORD
6335:  URLIS_URL','LongInt'( 0);
6336:  URLIS_OPAQUE','LongInt'( 1);
6337:  URLIS_NOHISTORY','LongInt'( 2);
```

```
6338:  URLIS_FILEURL','LongInt'( 3);
6339:  URLIS_APPLIABLE','LongInt'( 4);
6340:  URLIS_DIRECTORY','LongInt'( 5);
6341:  URLIS_HASQUERY','LongInt'( 6);
6342:  TUrlIs', 'DWORD
6343:  URL_UNESCAPE','LongWord( $10000000);
6344:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6345:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6346:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6347:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6348:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6349:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6350:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6351:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6352:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6353:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6354:  URL_INTERNAL_PATH','LongWord( $00800000);
6355:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6356:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6357:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6358:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6359:  URL_APPLY_DEFAULT','LongWord( $00000001);
6360:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6361:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6362:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6363:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6364:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6365:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6366:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6367:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6368:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6369:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6370:  Function UrlGetLocation( psz1 : PChar) : PChar
6371:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6372:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6373:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6374:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6375:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6376:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6377:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6378:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6379:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6380:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6381:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6382:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6383:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6384:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6385:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
      pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6386:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6387:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6388:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6389:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6390:  SHREGDEL_DEFAULT','LongWord( $00000000);
6391:  SHREGDEL_HKCU','LongWord( $00000001);
6392:  SHREGDEL_HKLM','LongWord( $00000010);
6393:  SHREGDEL_BOTH','LongWord( $00000011);
6394:  SHREGENUM_DEFAULT','LongWord( $00000000);
6395:  SHREGENUM_HKCU','LongWord( $00000001);
6396:  SHREGENUM_HKLM','LongWord( $00000010);
6397:  SHREGENUM_BOTH','LongWord( $00000011);
6398:  SHREGSET_HKCU','LongWord( $00000001);
6399:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6400:  SHREGSET_HKLM','LongWord( $00000004);
6401:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6402:  TSHRegDelFlags', 'DWORD
6403:  TSHRegEnumFlags', 'DWORD
6404:  HUSKEY', 'THandle
6405:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6406:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6407:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6408:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6409:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6410:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6411:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6412:  ASSOCF_VERIFY','LongWord( $00000040);
6413:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6414:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6415:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6416:  ASSOCF', 'DWORD
6417:  ASSOCSTR_COMMAND','LongInt'( 1);
6418:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6419:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6420:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6421:  ASSOCSTR_NOOPEN','LongInt'( 5);
6422:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6423:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6424:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6425:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
```

```
6426:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6427:  ASSOCSTR_INFOTIP','LongInt'( 11);
6428:  ASSOCSTR_MAX','LongInt'( 12);
6429:  ASSOCSTR', 'DWORD
6430:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6431:  ASSOCKEY_APP','LongInt'( 2);
6432:  ASSOCKEY_CLASS','LongInt'( 3);
6433:  ASSOCKEY_BASECLASS','LongInt'( 4);
6434:  ASSOCKEY_MAX','LongInt'( 5);
6435:  ASSOCKEY', 'DWORD
6436:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6437:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6438:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6439:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6440:  ASSOCDATA_MAX','LongInt'( 5);
6441:  ASSOCDATA', 'DWORD
6442:  ASSOCENUM_NONE','LongInt'( 0);
6443:  ASSOCENUM', 'DWORD
6444:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6445:  SHACF_DEFAULT $00000000);
6446:  SHACF_FILESYSTEM','LongWord( $00000001);
6447:  SHACF_URLHISTORY','LongWord( $00000002);
6448:  SHACF_URLMRU','LongWord( $00000004);
6449:  SHACF_USETAB','LongWord( $00000008);
6450:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6451:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6452:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6453:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6454:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6455:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6456:  Procedure SHSetThreadRef( punk : IUnknown)
6457:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6458:  CTF_INSIST','LongWord( $00000001);
6459:  CTF_THREAD_REF','LongWord( $00000002);
6460:  CTF_PROCESS_REF','LongWord( $00000004);
6461:  CTF_COINIT','LongWord( $00000008);
6462:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6463:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6464:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6465:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6466:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6467:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6468:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6469:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6470:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6471:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6472:  Function SetRectEmpty( var lprc : TRect) : BOOL
6473:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6474:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6475:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6476:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6477:
6478:  Function InitializeFlatSB( hWnd : HWND) : Bool
6479:  Procedure UninitializeFlatSB( hWnd : HWND)
6480:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6481:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6482:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6483:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6484:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6485:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6486:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6487:
6488:
6489: //  ****************************************** 204 unit uPSI_ShellAPI;
6490: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6491: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6492: Procedure DragFinish( Drop : HDROP)
6493: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6494: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6495: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6496: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6497: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6498: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6499: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6500: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6501: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6502: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6503: Procedure SHFreeNameMappings( hNameMappings : THandle)
6504:
6505:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6506:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6507:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6508:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6509:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6510:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6511:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6512:  Function SimpleXMLEncode( const S : string) : string
6513:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6514:  Function XMLEncode( const S : string) : string
```

```
6515:  Function XMLDecode( const S : string) : string
6516:  Function EntityEncode( const S : string) : string
6517:  Function EntityDecode( const S : string) : string
6518:
6519: procedure RIRegister_CPort_Routines(S: TPSExec);
6520:  Procedure EnumComPorts( Ports : TStrings)
6521:  Procedure ListComPorts( Ports : TStrings)
6522:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6523:  Function GetComPorts: TStringlist;
6524:  Function StrToBaudRate( Str : string) : TBaudRate
6525:  Function StrToStopBits( Str : string) : TStopBits
6526:  Function StrToDataBits( Str : string) : TDataBits
6527:  Function StrToParity( Str : string) : TParityBits
6528:  Function StrToFlowControl( Str : string) : TFlowControl
6529:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6530:  Function StopBitsToStr( StopBits : TStopBits) : string
6531:  Function DataBitsToStr( DataBits : TDataBits) : string
6532:  Function ParityToStr( Parity : TParityBits) : string
6533:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6534:  Function ComErrorsToStr( Errors : TComErrors) : String
6535:
6536:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6537:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6538:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6539:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6540:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6541:  Function GetMessagePos : DWORD
6542:  Function GetMessageTime : Longint
6543:  Function GetMessageExtraInfo : Longint
6544:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6545:  Procedure JAddToRecentDocs( const Filename : string)
6546:  Procedure ClearRecentDocs
6547:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6548:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6549:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6550:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6551:  Function RecycleFile( FileToRecycle : string) : Boolean
6552:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6553:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6554:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6555:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6556:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6557:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6558:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6559:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6560:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6561:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6562:
6563: ****************************************** unit uPSI_JclPeImage;
6564:
6565:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6566: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6567:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6568:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
6569:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6570:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6571:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6572:  Function PeDoesExportFunction(const FileName:TFileName;const
      FuncName:string;Options:TJclSmartCompOptions):Bool;
6573:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6574:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
      FunctionName:string;Options:TJclSmartCompOptions):Bool
6575:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean
6576:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
6577:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean
6578:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
      LibraryName:string; IncludeLibNames : Boolean): Boolean
6579:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6580:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6581:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6582:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
      NamesList:TStrings):Bool
6583:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6584:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
      Descript:Bool):Bool;
```

```
6585:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6586:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6587:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6588:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6589:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6590:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6591:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
      PImageSectionHeader
6592:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6593:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6594:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
      ___Pointer;
6595:    SIRegister_TJclPeSectionStream(CL);
6596:    SIRegister_TJclPeMapImgHookItem(CL);
6597:    SIRegister_TJclPeMapImgHooks(CL);
6598:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
      NtHeaders:TImageNtHeaders):Boolean
6599:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6600:    Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6601:    TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6602:    TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6603:    TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6604:    TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6605:    TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6606:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
      TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6607:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
      Descript:TJclBorUmDescription):TJclBorUmResult;
6608:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6609:  Function PeBorUnmangleName3( const Name : string) : string;
6610:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6611:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6612:
6613:
6614: //***************** SysTools uPSI_StSystem; ************************************
6615:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6616:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6617:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6618:  //Procedure EnumerateDirectories(const
      StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6619:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
      IncludeItem:TIncludeItemFunc);
6620:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6621:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6622:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6623:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6624:  Function FlushOsBuffers( Handle : Integer) : Boolean
6625:  Function GetCurrentUser : AnsiString
6626:  Function GetDiskClass( Drive : Char) : DiskClass
6627:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
      SectorsPerCluster:Cardinal):Bool;
6628:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
      DiskSize:Double):Bool;
6629:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
      DiskSize:Comp):Boolean;
6630:    { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6631:  Function getDiskSpace2(const path: String; index: integer): int64;
6632:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6633:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6634:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6635:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6636:  Function GetLongPath( const APath : AnsiString) : AnsiString
6637:  Function GetMachineName : AnsiString
6638:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6639:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6640:  Function GetShortPath( const APath : AnsiString) : AnsiString
6641:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6642:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6643:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6644:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6645:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6646:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6647:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6648:  Function IsDriveReady( Drive : Char) : Boolean
6649:  Function IsFile( const FileName : AnsiString) : Boolean
6650:  Function IsFileArchive( const S : AnsiString) : Integer
6651:  Function IsFileHidden( const S : AnsiString) : Integer
6652:  Function IsFileReadOnly( const S : AnsiString) : Integer
6653:  Function IsFileSystem( const S : AnsiString) : Integer
6654:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6655:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6656:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6657:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6658:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6659:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6660:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6661:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6662:  Function ValidDrive( Drive : Char) : Boolean
6663:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
```

```
6664:
6665: //****************************unit uPSI_JclLANMan;****************************************
6666: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6667:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6668:  Function DeleteAccount( const Servername, Username : string) : Boolean
6669:  Function DeleteLocalAccount( Username : string) : Boolean
6670:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6671:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6672:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6673:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6674:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6675:  Function LocalGroupExists( const Group : string) : Boolean
6676:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6677:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6678:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6679:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6680:  Function IsLocalAccount( const AccountName : string) : Boolean
6681:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6682:  Function GetRandomString( NumChar : cardinal) : string
6683:
6684: //****************************unit uPSI_cUtils;****************************************
6685: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6686: Function cIsWinNT : boolean
6687:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6688:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6689:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6690:  Function cGetShortName( FileName : string) : string
6691:  Procedure cShowError( Msg : String)
6692:  Function cCommaStrToStr( s : string; formatstr : string) : string
6693:  Function cIncludeQuoteIfSpaces( s : string) : string
6694:  Function cIncludeQuoteIfNeeded( s : string) : string
6695:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6696:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6697:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6698:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6699:  Function cCodeInstoStr( s : string) : string
6700:  Function cStrtoCodeIns( s : string) : string
6701:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6702:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6703:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6704:  Function cPointtoStr( const pt : TPoint) : string
6705:  Function cListtoStr( const List : TStrings) : string
6706:  Function ListtoStr( const List : TStrings) : string
6707:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6708:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6709:  Function cGetFileTyp( const FileName : string) : TUnitType
6710:  Function cGetExTyp( const FileName : string) : TExUnitType
6711:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6712:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6713:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6714:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6715:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6716:  Function cGenMakePath( FileName : String) : String;
6717:  Function cGenMakePath2( FileName : String) : String
6718:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6719:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6720:  Function cCalcMod( Count : Integer) : Integer
6721:  Function cGetVersionString( FileName : string) : string
6722:  Function cCheckChangeDir( var Dir : string) : boolean
6723:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6724:  Function cIsNumeric( s : string) : boolean
6725:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6726:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6727:  Function GetFileTyp( const FileName : string) : TUnitType
6728:  Function Atoi(const aStr: string): integer
6729:  Function Itoa(const aint: integer): string
6730:  Function Atof(const aStr: string): double');
6731:  Function Atol(const aStr: string): longint');
6732:
6733:
6734: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6735: begin
6736:   FindClass('TOBJECT'),'EHTTP
6737:   FindClass('TOBJECT'),'EHTTPParser
6738:   //AnsiCharSet', 'set of AnsiChar
6739:   AnsiStringArray', 'array of AnsiString
6740:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6741:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6742:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6743:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6744:    +'CustomMinVersion : Integer; end
6745:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6746:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6747:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6748:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
```

```
6749:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6750:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6751:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6752:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6753:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6754:    +'nection, hntOrigin, hntKeepAlive )
6755:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6756:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6757:    +' AnsiString; end
6758:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6759:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6760:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6761:   //PHTTPContentLength', '^THTTPContentLength // will not work
6762:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6763:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6764:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6765:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6766:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6767:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6768:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6769:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6770:    +' CustomStr : AnsiString; end
6771:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6772:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6773:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6774:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6775:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6776:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6777:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6778:    +'m; Custom : AnsiString; end
6779:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6780:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6781:    +' Custom : AnsiString; end
6782:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6783:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6784:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6785:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6786:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6787:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6788:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6789:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6790:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6791:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6792:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6793:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6794:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6795:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6796:    +'FieldEnum; List : array of THTTPContentEncoding; end
6797:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6798:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6799:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6800:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6801:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6802:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6803:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6804:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6805:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6806:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6807:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6808:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6809:    +'CustomFieldArray; Custom : AnsiString; end
6810:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6811:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6812:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6813:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6814:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6815:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6816:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6817:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6818:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6819:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6820:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6821:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6822:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6823:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6824:   THTTPFixedHeaders','array[0..42] of AnsiString
6825:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6826:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6827:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6828:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6829:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6830:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6831:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6832:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6833:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6834:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6835:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6836:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6837:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
```

```
6838:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6839:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6840:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6841:    +' THTTPDateField; Age : THTTPAgeField; end
6842:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6843:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6844:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6845:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6846:  Procedure InitHTTPRequest( var A : THTTPRequest)
6847:  Procedure InitHTTPResponse( var A : THTTPResponse)
6848:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6849:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6850:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6851:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6852:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6853:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6854:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6855:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6856:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6857:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6858:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6859:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6860:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6861:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6862:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6863:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6864:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6865:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6866:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6867:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6868:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6869:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6870:   THTTPStringOption', '( hsoNone )
6871:   THTTPStringOptions', 'set of THTTPStringOption
6872:   FindClass('TOBJECT'),'TAnsiStringBuilder
6873:
6874:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6875:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6876:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6877:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6878:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6879:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6880:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6881:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6882:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6883:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6884:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6885:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6886:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6887:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6888:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6889:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
6890:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6891:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6892:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions)
6893:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6894:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6895:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6896:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6897:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6898:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6899:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6900:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
```

```
6901:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6902:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
      THTTPStrOptions);
6903:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6904:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
      P:THTTPStringOptions);
6905:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6906:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6907:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6908:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6909:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6910:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6911:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
      Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6912:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6913:   SIRegister_THTTPParser(CL);
6914:   FindClass('TOBJECT'),'THTTPContentDecoder
6915:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6916:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6917:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6918:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6919:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6920:   SIRegister_THTTPContentDecoder(CL);
6921:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6922:   FindClass('TOBJECT'),'THTTPContentReader
6923:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6924:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
      LogLevel:Int;
6925:   SIRegister_THTTPContentReader(CL);
6926:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6927:   FindClass('TOBJECT'),'THTTPContentWriter
6928:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6929:   SIRegister_THTTPContentWriter(CL);
6930:  Procedure SelfTestcHTTPUtils
6931: end;
6932:
6933: (*----------------------------------------------------------------------------*)
6934: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6935: begin
6936:  'TLSLibraryVersion','String '1.00
6937:  'TLSError_None','LongInt'( 0);
6938:  'TLSError_InvalidBuffer','LongInt'( 1);
6939:  'TLSError_InvalidParameter','LongInt'( 2);
6940:  'TLSError_InvalidCertificate','LongInt'( 3);
6941:  'TLSError_InvalidState','LongInt'( 4);
6942:  'TLSError_DecodeError','LongInt'( 5);
6943:  'TLSError_BadProtocol','LongInt'( 6);
6944:  Function TLSErrorMessage( const TLSError : Integer) : String
6945:   SIRegister_ETLSError(CL);
6946:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6947:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6948:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6949:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6950:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6951:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6952:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6953:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6954:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6955:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6956:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6957:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6958:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6959:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6960:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6961:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6962:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6963:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6964:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6965:   PTLSRandom', '^TTLSRandom // will not work
6966:  Procedure InitTLSRandom( var Random : TTLSRandom)
6967:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6968:  'TLSSessionIDMaxLen','LongInt'( 32);
6969:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6970:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6971:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6972:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6973:    +'; Signature : TTLSSignatureAlgorithm; end
6974:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6975:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6976:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6977:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6978:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6979:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6980:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6981:    +'nteger; Supported : Boolean; end
6982:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6983:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
```

```
6984:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6985:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6986:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6987:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6988:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6989:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6990:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6991:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6992:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
6993:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
6994:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6995:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6996:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
6997:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6998:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6999:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
7000:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
7001:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
7002:    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
7003:    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
7004:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
      IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
7005:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
      ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
7006:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
7007:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
7008:  Procedure SelfTestcTLSUtils
7009: end;
7010:
7011: (*----------------------------------------------------------------------------*)
7012: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
7013: begin
7014:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
      integer; disks : integer; mx : integer; my : integer; end
7015:  // pBoard', '^tBoard // will not work
7016:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
7017:  Function rCheckMove( color : byte; cx, cy : integer) : integer
7018:  //Function rDoStep( data : pBoard) : word
7019:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
7020: end;
7021:
7022: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7023: begin
7024: Function InEditMode( ADataset : TDataset) : Boolean
7025: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
7026: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7027: Function GetFieldText( AField : TField) : String
7028: end;
7029:
7030: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
7031: begin
7032:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7033:   TMyPrintRange', '( prAll, prSelected )
7034:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7035:    +'ded, ssDateTime, ssTime, ssCustom )
7036:   TSortDirection', '( sdAscending, sdDescending )
7037:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7038:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7039:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7040:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7041:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7042:   SIRegister_TSortOptions(CL);
7043:   SIRegister_TPrintOptions(CL);
7044:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7045:   SIRegister_TSortedList(CL);
7046:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7047:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7048:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7049:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7050:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7051:   SIRegister_TFontSetting(CL);
7052:   SIRegister_TFontList(CL);
7053:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
7054:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7055:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7056:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7057:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
7058:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7059:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7060:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7061:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
7062:    +'r; var SortStyle : TSortStyle)
7063:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
```

```
7064:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7065:   SIRegister_TSortGrid(CL);
7066:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
7067:  Function NormalCompare( const Str1, Str2 : String) : Integer
7068:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
7069:  Function NumericCompare( const Str1, Str2 : String) : Integer
7070:  Function TimeCompare( const Str1, Str2 : String) : Integer
7071:  //Function Compare( Item1, Item2 : Pointer) : Integer
7072: end;
7073:
7074: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
7075:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
7076:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7077:  Procedure IBDataBaseError
7078:  Function StatusVector : PISC_STATUS
7079:  Function StatusVectorArray : PStatusVector
7080:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7081:  Function StatusVectorAsText : string
7082:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7083:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7084:
7085:
7086: //****************************unit uPSI_BoldUtils;****************************************
7087:  Function CharCount( c : char; const s : string) : integer
7088:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7089:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7090:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7091:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7092:  Function BoldTrim( const S : string) : string
7093:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7094:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7095:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7096:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7097:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7098:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7099:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7100:  Function CapitalisedToSpaced( Capitalised : String) : String
7101:  Function SpacedToCapitalised( Spaced : String) : String
7102:  Function BooleanToString( BoolValue : Boolean) : String
7103:  Function StringToBoolean( StrValue : String) : Boolean
7104:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7105:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7106:  Function StringListToVarArray( List : TStringList) : variant
7107:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7108:  Function GetComputerNameStr : string
7109:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7110:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7111:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7112:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7113:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7114:  Procedure EnsureTrailing( var Str : String; ch : char)
7115:  Function BoldDirectoryExists( const Name : string) : Boolean
7116:  Function BoldForceDirectories( Dir : string) : Boolean
7117:  Function BoldRootRegistryKey : string
7118:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7119:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7120:  Function LogicalAnd( A, B : Integer) : Boolean
7121:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7122:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7123:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7124:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7125:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7126:  Function IsFirstInstance : Boolean
7127:  Procedure ActivateFirst( AString : PChar)
7128:  Procedure ActivateFirstCommandLine
7129:  function MakeAckPkt(const BlockNumber: Word): string;
7130:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7131:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7132:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7133:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7134:  function IdStrToWord(const Value: String): Word;
7135:  function IdWordToStr(const Value: Word): WordStr;
7136:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7137:  Function CPUFeatures : TCPUFeatures
7138:
7139: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7140: begin
7141:   AddTypeS('TXRTLBitIndex', 'Integer
7142:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7143:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7144:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7145:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7146:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7147:  Function XRTLSwapHiLo16( X : Word) : Word
7148:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7149:  Function XRTLSwapHiLo64( X : Int64) : Int64
7150:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7151:  Function XRTLROR32( A, S : Cardinal) : Cardinal
```

```
7152:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7153:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7154:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7155:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7156:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7157:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7158:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7159:  Function XRTLPopulation( A : Cardinal) : Cardinal
7160: end;
7161:
7162: Function XRTLURLDecode( const ASrc : WideString) : WideString
7163: Function XRTLURLEncode( const ASrc : WideString) : WideString
7164: Function XRTLURINormalize( const AURI : WideString) : WideString
7165: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
     VPassword : WideString)
7166: Function XRTLExtractLongPathName(APath: string): string;
7167:
7168: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7169: begin
7170:   AddTypeS('Int8', 'ShortInt
7171:   AddTypeS('Int16', 'SmallInt
7172:   AddTypeS('Int32', 'LongInt
7173:   AddTypeS('UInt8', 'Byte
7174:   AddTypeS('UInt16', 'Word
7175:   AddTypeS('UInt32', 'LongWord
7176:   AddTypeS('UInt64', 'Int64
7177:   AddTypeS('Word8', 'UInt8
7178:   AddTypeS('Word16', 'UInt16
7179:   AddTypeS('Word32', 'UInt32
7180:   AddTypeS('Word64', 'UInt64
7181:   AddTypeS('LargeInt', 'Int64
7182:   AddTypeS('NativeInt', 'Integer
7183:   AddTypeS('NativeUInt', 'Cardinal
7184:   Const('BitsPerByte','LongInt'( 8);
7185:   Const('BitsPerWord','LongInt'( 16);
7186:   Const('BitsPerLongWord','LongInt'( 32);
7187:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7188:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7189:  Function MinI( const A, B : Integer) : Integer
7190:  Function MaxI( const A, B : Integer) : Integer
7191:  Function MinC( const A, B : Cardinal) : Cardinal
7192:  Function MaxC( const A, B : Cardinal) : Cardinal
7193:  Function SumClipI( const A, I : Integer) : Integer
7194:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7195:  Function InByteRange( const A : Int64) : Boolean
7196:  Function InWordRange( const A : Int64) : Boolean
7197:  Function InLongWordRange( const A : Int64) : Boolean
7198:  Function InShortIntRange( const A : Int64) : Boolean
7199:  Function InSmallIntRange( const A : Int64) : Boolean
7200:  Function InLongIntRange( const A : Int64) : Boolean
7201:   AddTypeS('Bool8', 'ByteBool
7202:   AddTypeS('Bool16', 'WordBool
7203:   AddTypeS('Bool32', 'LongBool
7204:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7205:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7206:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7207:  Const('MinSingle','Single').setExtended( 1.5E-45);
7208:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7209:  Const('MinDouble','Double').setExtended( 5.0E-324);
7210:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7211:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7212:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7213:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7214:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7215:  Function MinF( const A, B : Float) : Float
7216:  Function MaxF( const A, B : Float) : Float
7217:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7218:  Function InSingleRange( const A : Float) : Boolean
7219:  Function InDoubleRange( const A : Float) : Boolean
7220:  Function InCurrencyRange( const A : Float) : Boolean;
7221:  Function InCurrencyRange1( const A : Int64) : Boolean;
7222:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7223:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7224:  Function FloatIsInfinity( const A : Extended) : Boolean
7225:  Function FloatIsNaN( const A : Extended) : Boolean
7226:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7227:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7228:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7229:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7230:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7231:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7232:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7233:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7234:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7235:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7236:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7237:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7238:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7239:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
```

```
7240:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7241:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7242:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7243:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7244:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7245:  Function SetBitScanForward( const Value : LongWord) : Integer;
7246:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7247:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7248:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7249:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7250:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7251:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7252:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7253:  Function cReverseBits( const Value : LongWord) : LongWord;
7254:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7255:  Function cSwapEndian( const Value : LongWord) : LongWord
7256:  Function cTwosComplement( const Value : LongWord) : LongWord
7257:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7258:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7259:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7260:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7261:  Function cBitCount( const Value : LongWord) : LongWord
7262:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7263:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7264:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7265:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7266:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7267:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7268:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7269:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7270:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7271: //  AddTypeS('CharSet', 'set of AnsiChar
7272:   AddTypeS('CharSet', 'set of Char       //!!!
7273:   AddTypeS('AnsiCharSet', 'TCharSet
7274:   AddTypeS('ByteSet', 'set of Byte
7275:   AddTypeS('AnsiChar', 'Char
7276:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7277:  Function AsByteSet( const C : array of Byte) : ByteSet
7278:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7279:  Procedure ClearCharSet( var C : CharSet)
7280:  Procedure FillCharSet( var C : CharSet)
7281:  procedure FillCharSearchRec;  // with 0
7282:  Procedure ComplementCharSet( var C : CharSet)
7283:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7284:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7285:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7286:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7287:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7288:  Function IsSubSet( const A, B : CharSet) : Boolean
7289:  Function IsEqual( const A, B : CharSet) : Boolean
7290:  Function IsEmpty( const C : CharSet) : Boolean
7291:  Function IsComplete( const C : CharSet) : Boolean
7292:  Function cCharCount( const C : CharSet) : Integer
7293:  Procedure ConvertCaseInsensitive( var C : CharSet)
7294:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7295:  Function IntRangeLength( const Low, High : Integer) : Int64
7296:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7297:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7298:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7299:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7300:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7301:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7302:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7303:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7304:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7305:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7306:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7307:   AddTypeS('UnicodeChar', 'WideChar
7308:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7309:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7310:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7311:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7312:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7313:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7314:  Function cSgn( const A : LongInt) : Integer;
7315:  Function cSgn1( const A : Int64) : Integer;
7316:  Function cSgn2( const A : Extended) : Integer;
7317:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7318:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7319:  Function WideCharToInt( const A : WideChar) : Integer
7320:  Function CharToInt( const A : Char) : Integer
7321:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7322:  Function IntToWideChar( const A : Integer) : WideChar
7323:  Function IntToChar( const A : Integer) : Char
7324:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7325:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7326:  Function IsHexChar( const Ch : Char) : Boolean
7327:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7328:  Function HexWideCharToInt( const A : WideChar) : Integer
```

```
7329:  Function HexCharToInt( const A : Char) : Integer
7330:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7331:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7332:  Function IntToUpperHexChar( const A : Integer) : Char
7333:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7334:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7335:  Function IntToLowerHexChar( const A : Integer) : Char
7336:  Function IntToStringA( const A : Int64) : AnsiString
7337:  Function IntToStringW( const A : Int64) : WideString
7338:  Function IntToString( const A : Int64) : String
7339:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7340:  Function UIntToStringW( const A : NativeUInt) : WideString
7341:  Function UIntToString( const A : NativeUInt) : String
7342:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7343:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7344:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7345:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7346:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7347:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7348:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7349:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7350:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7351:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7352:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7353:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7354:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7355:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7356:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7357:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7358:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7359:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7360:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7361:  Function StringToInt64A( const S : AnsiString) : Int64
7362:  Function StringToInt64W( const S : WideString) : Int64
7363:  Function StringToInt64( const S : String) : Int64
7364:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7365:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7366:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7367:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7368:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7369:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7370:  Function StringToIntA( const S : AnsiString) : Integer
7371:  Function StringToIntW( const S : WideString) : Integer
7372:  Function StringToInt( const S : String) : Integer
7373:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7374:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7375:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7376:  Function StringToLongWordA( const S : AnsiString) : LongWord
7377:  Function StringToLongWordW( const S : WideString) : LongWord
7378:  Function StringToLongWord( const S : String) : LongWord
7379:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7380:  Function HexToUIntW( const S : WideString) : NativeUInt
7381:  Function HexToUInt( const S : String) : NativeUInt
7382:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7383:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7384:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7385:  Function HexToLongWordA( const S : AnsiString) : LongWord
7386:  Function HexToLongWordW( const S : WideString) : LongWord
7387:  Function HexToLongWord( const S : String) : LongWord
7388:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7389:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7390:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7391:  Function OctToLongWordA( const S : AnsiString) : LongWord
7392:  Function OctToLongWordW( const S : WideString) : LongWord
7393:  Function OctToLongWord( const S : String) : LongWord
7394:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7395:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7396:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7397:  Function BinToLongWordA( const S : AnsiString) : LongWord
7398:  Function BinToLongWordW( const S : WideString) : LongWord
7399:  Function BinToLongWord( const S : String) : LongWord
7400:  Function FloatToStringA( const A : Extended) : AnsiString
7401:  Function FloatToStringW( const A : Extended) : WideString
7402:  Function FloatToString( const A : Extended) : String
7403:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7404:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7405:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7406:  Function StringToFloatA( const A : AnsiString) : Extended
7407:  Function StringToFloatW( const A : WideString) : Extended
7408:  Function StringToFloat( const A : String) : Extended
7409:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7410:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7411:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7412:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
      PadChar: AnsiChar) : AnsiString
7413:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7414:  unit uPSI_cFundamentUtils;
7415:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7416:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
```

```
7417:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7418:  Const('CCHARSET','Stringb64_XXEncode);
7419:  Const('CHEXSET','String'0123456789ABCDEF
7420:  Const('HEXDIGITS','String'0123456789ABCDEF
7421:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7422:  Const('DIGISET','String'0123456789
7423:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7424:  Const('DIGISET2','TCharset').SetSet('0123456789'
7425:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7426:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7427:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7428:  Const('NUMBERS','String'0123456789');
7429:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7430:  Function CharSetToStr( const C : CharSet) : AnsiString
7431:  Function StrToCharSet( const S : AnsiString) : CharSet
7432:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7433:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7434:  Function UUDecode( const S : AnsiString) : AnsiString
7435:  Function XXDecode( const S : AnsiString) : AnsiString
7436:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7437:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7438:  Function InterfaceToStrW( const I : IInterface) : WideString
7439:  Function InterfaceToStr( const I : IInterface) : String
7440:  Function ObjectClassName( const O : TObject) : String
7441:  Function ClassClassName( const C : TClass) : String
7442:  Function ObjectToStr( const O : TObject) : String
7443:  Function ObjectToString( const O : TObject) : String
7444:  Function CharSetToStr( const C : CharSet) : AnsiString
7445:  Function StrToCharSet( const S : AnsiString) : CharSet
7446:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7447:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7448:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7449:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7450:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7451:  Const('Bytes1KB','LongInt'( 1024);
7452:   SIRegister_IInterface(CL);
7453:  Procedure SelfTestCFundamentUtils
7454:
7455: Function CreateSchedule : IJclSchedule
7456: Function NullStamp : TTimeStamp
7457: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7458: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7459: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7460:
7461: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7462: begin
7463:  AddTypeS('TFunc', 'function(X : Float) : Float;
7464: Function InitGraphics( Width, Height : Integer) : Boolean
7465: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7466: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7467: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7468: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7469: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7470: Procedure SetGraphTitle( Title : String)
7471: Procedure SetOxTitle( Title : String)
7472: Procedure SetOyTitle( Title : String)
7473: Function GetGraphTitle : String
7474: Function GetOxTitle : String
7475: Function GetOyTitle : String
7476: Procedure PlotOxAxis( Canvas : TCanvas)
7477: Procedure PlotOyAxis( Canvas : TCanvas)
7478: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7479: Procedure WriteGraphTitle( Canvas : TCanvas)
7480: Function SetMaxCurv( NCurv : Byte) : Boolean
7481: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7482: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7483: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7484: Procedure SetCurvStep( CurvIndex, Step : Integer)
7485: Function GetMaxCurv : Byte
7486: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7487: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7488: Function GetCurvLegend( CurvIndex : Integer) : String
7489: Function GetCurvStep( CurvIndex : Integer) : Integer
7490: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7491: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7492: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7493: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7494: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7495: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7496: Function Xpixel( X : Float) : Integer
7497: Function Ypixel( Y : Float) : Integer
7498: Function Xuser( X : Integer) : Float
7499: Function Yuser( Y : Integer) : Float
7500: end;
7501:
7502: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
```

```
7503: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7504: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7505: Procedure FFT_Integer_Cleanup
7506: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7507: //unit uPSI_JclStreams;
7508: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7509: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7510: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7511: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7512:
7513: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7514: begin
7515:  FindClass('TOBJECT'),'EInvalidDest
7516:  FindClass('TOBJECT'),'EFCantMove
7517:  Procedure fmxCopyFile( const FileName, DestName : string)
7518:  Procedure fmxMoveFile( const FileName, DestName : string)
7519:  Function fmxGetFileSize( const FileName : string) : LongInt
7520:  Function fmxFileDateTime( const FileName : string) : TDateTime
7521:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7522:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7523: end;
7524:
7525: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7526: begin
7527:  SIRegister_IFindFileIterator(CL);
7528:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7529: end;
7530:
7531: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7532: begin
7533:  Function SkipWhite( cp : PChar) : PChar
7534:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7535:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7536:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7537: end;
7538:
7539: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7540: begin
7541:   SIRegister_TStringHashMapTraits(CL);
7542:  Function CaseSensitiveTraits : TStringHashMapTraits
7543:  Function CaseInsensitiveTraits : TStringHashMapTraits
7544:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7545:    +'e; Right : PHashNode; end
7546:   //PHashArray', '^THashArray // will not work
7547:   SIRegister_TStringHashMap(CL);
7548:  THashValue', 'Cardinal
7549:  Function StrHash( const s : string) : THashValue
7550:  Function TextHash( const s : string) : THashValue
7551:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7552:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7553:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7554:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7555:   SIRegister_TCaseSensitiveTraits(CL);
7556:   SIRegister_TCaseInsensitiveTraits(CL);
7557:
7558:
7559: //************************************************************************unit uPSI_umath;
7560:  Function uExpo( X : Float) : Float
7561:  Function uExp2( X : Float) : Float
7562:  Function uExp10( X : Float) : Float
7563:  Function uLog( X : Float) : Float
7564:  Function uLog2( X : Float) : Float
7565:  Function uLog10( X : Float) : Float
7566:  Function uLogA( X, A : Float) : Float
7567:  Function uIntPower( X : Float; N : Integer): Float
7568:  Function uPower( X, Y : Float) : Float
7569:  Function SgnGamma( X : Float) : Integer
7570:  Function Stirling( X : Float) : Float
7571:  Function StirLog( X : Float) : Float
7572:  Function Gamma( X : Float) : Float
7573:  Function LnGamma( X : Float) : Float
7574:  Function DiGamma( X : Float) : Float
7575:  Function TriGamma( X : Float) : Float
7576:  Function IGamma( X : Float) : Float
7577:  Function JGamma( X : Float) : Float
7578:  Function InvGamma( X : Float) : Float
7579:  Function Erf( X : Float) : Float
7580:  Function Erfc( X : Float) : Float
7581: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7582: { Correlation coefficient between samples X and Y }
7583: function DBeta(A, B, X : Float) : Float;
7584: { Density of Beta distribution with parameters A and B }
7585: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7586: Function Beta(X, Y : Float) : Float
7587: Function Binomial( N, K : Integer) : Float
7588: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7589: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7590: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7591: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
```

```
7592: Function DNorm( X : Float) : Float
7593:
7594: function DGamma(A, B, X : Float) : Float;
7595: { Density of Gamma distribution with parameters A and B }
7596: function DKhi2(Nu : Integer; X : Float) : Float;
7597: { Density of Khi-2 distribution with Nu d.o.f. }
7598: function DStudent(Nu : Integer; X : Float) : Float;
7599: { Density of Student distribution with Nu d.o.f. }
7600: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7601: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7602: function IBeta(A, B, X : Float) : Float;
7603: { Incomplete Beta function}
7604: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7605:
7606: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7607: begin
7608:  Procedure SetOptAlgo( Algo : TOptAlgo)
7609: procedure SetOptAlgo(Algo : TOptAlgo);
7610: { ----------------------------------------------------------------
7611:   Sets the optimization algorithm according to Algo, which must be
7612:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7613:
7614:  Function GetOptAlgo : TOptAlgo
7615:  Procedure SetMaxParam( N : Byte)
7616:  Function GetMaxParam : Byte
7617:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7618:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7619:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7620:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
     Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7621:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
     MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7622:  Procedure SetMCFile( FileName : String)
7623:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7624:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
     LastPar:Integer;V:TMatrix);
7625: end;
7626:
7627: (*----------------------------------------------------------------------------*)
7628: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7629: begin
7630:  Procedure SaveSimplex( FileName : string)
7631:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7632: end;
7633: (*----------------------------------------------------------------------------*)
7634: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7635: begin
7636:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7637:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7638: end;
7639:
7640: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7641: begin
7642:  Function LTrim( S : String) : String
7643:  Function RTrim( S : String) : String
7644:  Function uTrim( S : String) : String
7645:  Function StrChar( N : Byte; C : Char) : String
7646:  Function RFill( S : String; L : Byte) : String
7647:  Function LFill( S : String; L : Byte) : String
7648:  Function CFill( S : String; L : Byte) : String
7649:  Function Replace( S : String; C1, C2 : Char) : String
7650:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7651:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7652:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7653:  Function FloatStr( X : Float) : String
7654:  Function IntStr( N : LongInt) : String
7655:  Function uCompStr( Z : Complex) : String
7656: end;
7657:
7658: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7659: begin
7660:  Function uSinh( X : Float) : Float
7661:  Function uCosh( X : Float) : Float
7662:  Function uTanh( X : Float) : Float
7663:  Function uArcSinh( X : Float) : Float
7664:  Function uArcCosh( X : Float) : Float
7665:  Function ArcTanh( X : Float) : Float
7666:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7667: end;
7668:
7669: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7670: begin
7671: type RNG_Type =
7672:   (RNG_MWC,      { Multiply-With-Carry }
7673:    RNG_MT,       { Mersenne Twister }
7674:    RNG_UVAG);    { Universal Virtual Array Generator }
7675:  Procedure SetRNG( RNG : RNG_Type)
7676:  Procedure InitGen( Seed : RNG_IntType)
7677:  Procedure SRand( Seed : RNG_IntType)
```

```
7678:  Function IRanGen : RNG_IntType
7679:  Function IRanGen31 : RNG_IntType
7680:  Function RanGen1 : Float
7681:  Function RanGen2 : Float
7682:  Function RanGen3 : Float
7683:  Function RanGen53 : Float
7684: end;
7685:
7686: //  Optimization by Simulated Annealing
7687: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7688: begin
7689:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7690:  Procedure SA_CreateLogFile( FileName : String)
7691:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7692: end;
7693:
7694: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7695: begin
7696:  Procedure InitUVAGbyString( KeyPhrase : string)
7697:  Procedure InitUVAG( Seed : RNG_IntType)
7698:  Function IRanUVAG : RNG_IntType
7699: end;
7700:
7701: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7702: begin
7703:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7704:  Procedure GA_CreateLogFile( LogFileName : String)
7705:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7706: end;
7707:
7708:  TVector', 'array of Float
7709: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7710: begin
7711:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7712:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7713: end;
7714:
7715: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7716: begin
7717:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7718:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7719: end;
7720:
7721: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7722: begin
7723:   FT_Result', 'Integer
7724:   //TDWordptr', '^DWord // will not work
7725:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7726:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7727:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7728:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7729:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7730:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7731:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7732:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7733:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7734:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7735:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7736:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7737:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7738:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7739:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7740:    yte; end
7741: end;
7742:
7743:
7744: //************************************ PaintFX*****************************
7745: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7746: begin
7747:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7748:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7749:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7750:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7751:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7752:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7753:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7754:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7755:     Procedure Turn( Src, Dst : TBitmap)
7756:     Procedure TurnRight( Src, Dst : TBitmap)
7757:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7758:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7759:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7760:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7761:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7762:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7763:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7764:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7765:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7766:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
```

```
7767:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7768:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7769:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7770:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7771:     Procedure Emboss( var Bmp : TBitmap)
7772:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7773:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7774:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7775:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7776:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7777:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7778:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7779:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7780:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7781:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7782:     Procedure SemiOpaque( Src, Dst : TBitmap)
7783:     Procedure ShadowDownLeft( const Dst : TBitmap)
7784:     Procedure ShadowDownRight( const Dst : TBitmap)
7785:     Procedure ShadowUpLeft( const Dst : TBitmap)
7786:     Procedure ShadowUpRight( const Dst : TBitmap)
7787:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7788:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7789:     Procedure FlipRight( const Dst : TBitmap)
7790:     Procedure FlipDown( const Dst : TBitmap)
7791:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7792:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7793:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7794:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7795:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7796:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7797:     Procedure SmoothResize( var Src, Dst : TBitmap)
7798:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7799:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7800:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7801:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7802:     Procedure GrayScale( const Dst : TBitmap)
7803:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7804:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7805:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7806:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7807:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7808:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7809:     Procedure AntiAlias( const Dst : TBitmap)
7810:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7811:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7812:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7813:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7814:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7815:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7816:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7817:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7818:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7819:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7820:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7821:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7822:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7823:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7824:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7825:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7826:     Procedure Invert( Src : TBitmap)
7827:     Procedure MirrorRight( Src : TBitmap)
7828:     Procedure MirrorDown( Src : TBitmap)
7829:   end;
7830: end;
7831:
7832: (*----------------------------------------------------------------------------*)
7833: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7834: begin
7835:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7836:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7837:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7838:   SIRegister_TJvPaintFX(CL);
7839:  Function SplineFilter( Value : Single) : Single
7840:  Function BellFilter( Value : Single) : Single
7841:  Function TriangleFilter( Value : Single) : Single
7842:  Function BoxFilter( Value : Single) : Single
7843:  Function HermiteFilter( Value : Single) : Single
7844:  Function Lanczos3Filter( Value : Single) : Single
7845:  Function MitchellFilter( Value : Single) : Single
7846: end;
7847:
7848:
7849: (*----------------------------------------------------------------------------*)
7850: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7851: begin
7852:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7853:  TeeMsg_DefaultSeriesName','String 'Series
7854:  TeeMsg_DefaultToolName','String 'ChartTool
7855:  ChartComponentPalette','String 'TeeChart
```

```
7856:  TeeMaxLegendColumns',LongInt'( 2);
7857:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7858:  TeeTitleFootDistance,LongInt( 5);
7859:   SIRegister_TCustomChartWall(CL);
7860:   SIRegister_TChartWall(CL);
7861:   SIRegister_TChartLegendGradient(CL);
7862:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7863:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7864:   FindClass('TOBJECT'),'LegendException
7865:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7866:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7867:   FindClass('TOBJECT'),'TCustomChartLegend
7868:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7869:   TLegendSymbolPosition', '( spLeft, spRight )
7870:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7871:   TSymbolCalcHeight', 'Function  : Integer
7872:   SIRegister_TLegendSymbol(CL);
7873:   SIRegister_TTeeCustomShapePosition(CL);
7874:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7875:   SIRegister_TLegendTitle(CL);
7876:   SIRegister_TLegendItem(CL);
7877:   SIRegister_TLegendItems(CL);
7878:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7879:   FindClass('TOBJECT'),'TCustomChart
7880:   SIRegister_TCustomChartLegend(CL);
7881:   SIRegister_TChartLegend(CL);
7882:   SIRegister_TChartTitle(CL);
7883:   SIRegister_TChartFootTitle(CL);
7884:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7885:    +'eButton; Shift : TShiftState; X, Y : Integer)
7886:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7887:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7888:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7889:    +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7890:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7891:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7892:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7893:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7894:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7895:    +'toMax : Boolean; Min : Double; Max : Double; end
7896:   TAllAxisSavedScales', 'array of TAxisSavedScales
7897:   SIRegister_TChartBackWall(CL);
7898:   SIRegister_TChartRightWall(CL);
7899:   SIRegister_TChartBottomWall(CL);
7900:   SIRegister_TChartLeftWall(CL);
7901:   SIRegister_TChartWalls(CL);
7902:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7903:   SIRegister_TCustomChart(CL);
7904:   SIRegister_TChart(CL);
7905:   SIRegister_TTeeSeriesTypes(CL);
7906:   SIRegister_TTeeToolTypes(CL);
7907:   SIRegister_TTeeDragObject(CL);
7908:   SIRegister_TColorPalettes(CL);
7909:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7910:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7911:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7912:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7913:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7914:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7915:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7916:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7917:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7918:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7919:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7920:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7921:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7922:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7923:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7924:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7925:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7926:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7927:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7928:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7929:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
      R:TRect;ReferenceChart:TCustomChart);
7930:   SIRegister_TChartTheme(CL);
7931:   //TChartThemeClass', 'class of TChartTheme
7932:   //TCanvasClass', 'class of TCanvas3D
7933:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7934:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7935:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7936:  Procedure ShowMessageUser( const S : String)
7937:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7938:  Function HasLabels( ASeries : TChartSeries) : Boolean
7939:  Function HasColors( ASeries : TChartSeries) : Boolean
```

```
7940:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7941:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7942: end;
7943:
7944:
7945: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7946: begin
7947:  //'TeeFormBorderStyle',' bsNone);
7948:   SIRegister_TMetafile(CL);
7949:  'TeeDefVerticalMargin','LongInt'( 4);
7950:  'TeeDefHorizMargin','LongInt'( 3);
7951:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7952:  'TeeMsg_TeeHand','String 'crTeeHand
7953:  'TeeNormalPrintDetail','LongInt'( 0);
7954:  'TeeHighPrintDetail','LongInt'( - 100);
7955:  'TeeDefault_PrintMargin','LongInt'( 15);
7956:  'MaxDefaultColors','LongInt'( 19);
7957:  'TeeTabDelimiter','Char #9);
7958:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7959:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7960:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7961:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7962:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7963:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7964:   SIRegister_TCustomPanelNoCaption(CL);
7965:   FindClass('TOBJECT'),'TCustomTeePanel
7966:   SIRegister_TZoomPanning(CL);
7967:   SIRegister_TTeeEvent(CL);
7968:   //SIRegister_TTeeEventListeners(CL);
7969:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7970:   SIRegister_TTeeMouseEvent(CL);
7971:   SIRegister_TCustomTeePanel(CL);
7972:   //TChartGradient', 'TTeeGradient
7973:   //TChartGradientClass', 'class of TChartGradient
7974:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7975:   SIRegister_TTeeZoomPen(CL);
7976:   SIRegister_TTeeZoomBrush(CL);
7977:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7978:   SIRegister_TTeeZoom(CL);
7979:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7980:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7981:   SIRegister_TBackImage(CL);
7982:   SIRegister_TCustomTeePanelExtended(CL);
7983:   //TChartBrushClass', 'class of TChartBrush
7984:   SIRegister_TTeeCustomShapeBrushPen(CL);
7985:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7986:   TTextFormat', '( ttfNormal, ttfHtml )
7987:   SIRegister_TTeeCustomShape(CL);
7988:   SIRegister_TTeeShape(CL);
7989:   SIRegister_TTeeExportData(CL);
7990:  Function TeeStr( const Num : Integer) : String
7991:  Function DateTimeDefaultFormat( const AStep : Double) : String
7992:  Function TEEDaysInMonth( Year, Month : Word) : Word
7993:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7994:  Function NextDateTimeStep( const AStep : Double) : Double
7995:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7996:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7997:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7998:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7999:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
8000:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
8001:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
8002:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
8003:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
8004:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
8005:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
8006:  Function DelphiToLocalFormat( const Format : String) : String
8007:  Function LocalToDelphiFormat( const Format : String) : String
8008:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
8009:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
8010:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
8011:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
8012:   TTeeSortSwap', 'Procedure ( a, b : Integer)
8013:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8014:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
8015:  Function TeeExtractField( St : String; Index : Integer) : String;
8016:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
8017:  Function TeeNumFields( St : String) : Integer;
8018:  Function TeeNumFields1( const St, Separator : String) : Integer;
8019:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8020:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
8021:  // TColorArray', 'array of TColor
8022:  Function GetDefaultColor( const Index : Integer) : TColor
8023:  Procedure SetDefaultColorPalette;
8024:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
8025:  'TeeCheckBoxSize','LongInt'( 11);
8026:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8027:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
```

```
8028:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8029:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8030:  Procedure TeeTranslateControl( AControl : TControl);
8031:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
8032:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8033:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8034:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8035:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8036:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8037:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8038:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8039:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8040:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8041:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8042:  Procedure TeeSaveStringOption( const AKey, Value : String)
8043:  Function TeeDefaultXMLEncoding : String
8044:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8045:   TeeWindowHandle', 'Integer
8046:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
8047:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8048:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8049: end;
8050:
8051:
8052: using mXBDEUtils
8053: ********************************************************************************
8054:  Procedure SetAlias( aAlias, aDirectory : String)
8055:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
8056:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
8057:  Procedure SetBDE( aPath, aNode, aValue : String)
8058:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8059:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8060:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8061:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8062:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8063:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8064:
8065:
8066: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8067: begin
8068: AddClassN(FindClass('TOBJECT'),'EDateTime
8069: Function DatePart( const D : TDateTime) : Integer
8070: Function TimePart( const D : TDateTime) : Double
8071: Function Century( const D : TDateTime) : Word
8072: Function Year( const D : TDateTime) : Word
8073: Function Month( const D : TDateTime) : Word
8074: Function Day( const D : TDateTime) : Word
8075: Function Hour( const D : TDateTime) : Word
8076: Function Minute( const D : TDateTime) : Word
8077: Function Second( const D : TDateTime) : Word
8078: Function Millisecond( const D : TDateTime) : Word
8079: ('OneDay','Extended').setExtended( 1.0);
8080: ('OneHour','Extended').SetExtended( OneDay / 24);
8081: ('OneMinute','Extended').SetExtended( OneHour / 60);
8082: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8083: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8084: ('OneWeek','Extended').SetExtended( OneDay * 7);
8085: ('HoursPerDay','Extended').SetExtended( 24);
8086: ('MinutesPerHour','Extended').SetExtended( 60);
8087: ('SecondsPerMinute','Extended').SetExtended( 60);
8088: Procedure SetYear( var D : TDateTime; const Year : Word)
8089: Procedure SetMonth( var D : TDateTime; const Month : Word)
8090: Procedure SetDay( var D : TDateTime; const Day : Word)
8091: Procedure SetHour( var D : TDateTime; const Hour : Word)
8092: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8093: Procedure SetSecond( var D : TDateTime; const Second : Word)
8094: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8095: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8096: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8097: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8098: Function IsAM( const D : TDateTime) : Boolean
8099: Function IsPM( const D : TDateTime) : Boolean
8100: Function IsMidnight( const D : TDateTime) : Boolean
8101: Function IsNoon( const D : TDateTime) : Boolean
8102: Function IsSunday( const D : TDateTime) : Boolean
8103: Function IsMonday( const D : TDateTime) : Boolean
8104: Function IsTuesday( const D : TDateTime) : Boolean
8105: Function IsWedneday( const D : TDateTime) : Boolean
8106: Function IsThursday( const D : TDateTime) : Boolean
8107: Function IsFriday( const D : TDateTime) : Boolean
8108: Function IsSaturday( const D : TDateTime) : Boolean
8109: Function IsWeekend( const D : TDateTime) : Boolean
8110: Function Noon( const D : TDateTime) : TDateTime
8111: Function Midnight( const D : TDateTime) : TDateTime
8112: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8113: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8114: Function NextWorkday( const D : TDateTime) : TDateTime
8115: Function PreviousWorkday( const D : TDateTime) : TDateTime
```

```
8116: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8117: Function LastDayOfYear( const D : TDateTime) : TDateTime
8118: Function EasterSunday( const Year : Word) : TDateTime
8119: Function GoodFriday( const Year : Word) : TDateTime
8120: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8121: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8122: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8123: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8124: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8125: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8126: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8127: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8128: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8129: Function DayOfYear( const D : TDateTime) : Integer
8130: Function DaysInMonth( const Ye, Mo : Word) : Integer
8131: Function DaysInMonth( const D : TDateTime) : Integer
8132: Function DaysInYear( const Ye : Word) : Integer
8133: Function DaysInYearDate( const D : TDateTime) : Integer
8134: Function WeekNumber( const D : TDateTime) : Integer
8135: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8136: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8137: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8138: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8139: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8140: Function DiffHours( const D1, D2 : TDateTime) : Integer
8141: Function DiffDays( const D1, D2 : TDateTime) : Integer
8142: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8143: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8144: Function DiffYears( const D1, D2 : TDateTime) : Integer
8145: Function GMTBias : Integer
8146: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8147: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8148: Function NowAsGMTTime : TDateTime
8149: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8150: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8151: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8152: Function DateTimeToANSI( const D : TDateTime) : Integer
8153: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8154: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8155: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8156: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8157: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8158: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8159: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8160: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8161: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8162: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8163: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8164: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8165: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8166: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8167: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8168: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8169: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8170: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8171: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8172: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8173: Function EnglishShortMonthA( const S : AnsiString) : Integer
8174: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8175: Function EnglishLongMonthA( const S : AnsiString) : Integer
8176: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8177: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8178: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8179: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8180: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8181: Function RFCMonthA( const S : AnsiString) : Word
8182: Function RFCMonthU( const S : UnicodeString) : Word
8183: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8184: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8185: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8186: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8187: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8188: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8189: Function NowAsRFCDateTimeA : AnsiString
8190: Function NowAsRFCDateTimeU : UnicodeString
8191: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8192: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8193: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8194: Function TimePeriodStr( const D : TDateTime) : AnsiString
8195: Procedure SelfTest
8196: end;
8197: //****************************************CFileUtils
8198: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8199: Function PathHasDriveLetter( const Path : String) : Boolean
8200: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8201: Function PathIsDriveLetter( const Path : String) : Boolean
8202: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8203: Function PathIsDriveRoot( const Path : String) : Boolean
8204: Function PathIsRootA( const Path : AnsiString) : Boolean
```

```
8205: Function PathIsRoot( const Path : String) : Boolean
8206: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8207: Function PathIsUNCPath( const Path : String) : Boolean
8208: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8209: Function PathIsAbsolute( const Path : String) : Boolean
8210: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8211: Function PathIsDirectory( const Path : String) : Boolean
8212: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8213: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8214: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8215: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8216: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8217: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8218: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8219: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8220: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8221: Function PathCanonical( const Path : String; const PathSep : Char) : String
8222: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8223: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8224: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8225: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8226: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8227: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8228: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8229: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8230: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8231: Function FileNameValid( const FileName : String) : String
8232: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8233: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8234: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8235: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8236: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8237: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8238: Procedure CCopyFile( const FileName, DestName : String)
8239: Procedure CMoveFile( const FileName, DestName : String)
8240: Function CDeleteFiles( const FileMask : String) : Boolean
8241: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8242: Procedure FileCloseEx( const FileHandle : TFileHandle)
8243: Function FileExistsA( const FileName : AnsiString) : Boolean
8244:  Function CFileExists( const FileName : String) : Boolean
8245:  Function CFileGetSize( const FileName : String) : Int64
8246: Function FileGetDateTime( const FileName : String) : TDateTime
8247: Function FileGetDateTime2( const FileName : String) : TDateTime
8248: Function FileIsReadOnly( const FileName : String) : Boolean
8249: Procedure FileDeleteEx( const FileName : String)
8250: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8251: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
     : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8252: Function DirectoryEntryExists( const Name : String) : Boolean
8253: Function DirectoryEntrySize( const Name : String) : Int64
8254:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8255: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8256: Procedure CDirectoryCreate( const DirectoryName : String)
8257: Function GetFirstFileNameMatching( const FileMask : String) : String
8258: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8259: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8260: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8261:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8262:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8263: Function DriveIsValid( const Drive : Char) : Boolean
8264: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8265: Function DriveFreeSpace( const Path : AnsiString) : Int64
8266:
8267: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8268: begin
8269:  AddClassN(FindClass('TOBJECT'),'ETimers
8270:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8271: Function TickDelta( const D1, D2 : LongWord) : Integer
8272: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8273:   AddTypeS('THPTimer', 'Int64
8274: Procedure StartTimer( var Timer : THPTimer)
8275: Procedure StopTimer( var Timer : THPTimer)
8276: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8277: Procedure InitStoppedTimer( var Timer : THPTimer)
8278: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8279: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8280: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8281: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8282: Function GetHighPrecisionFrequency : Int64
8283: Function GetHighPrecisionTimerOverhead : Int64
8284: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8285: Procedure SelfTestCTimer
8286: end;
8287:
8288: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8289: begin
8290:  Function RandomSeed : LongWord
8291:  Procedure AddEntropy( const Value : LongWord)
8292:  Function RandomUniform : LongWord;
```

```
8293:  Function RandomUniform1( const N : Integer) : Integer;
8294:  Function RandomBoolean : Boolean
8295:  Function RandomByte : Byte
8296:  Function RandomByteNonZero : Byte
8297:  Function RandomWord : Word
8298:  Function RandomInt64 : Int64;
8299:  Function RandomInt641( const N : Int64) : Int64;
8300:  Function RandomHex( const Digits : Integer) : String
8301:  Function RandomFloat : Extended
8302:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8303:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8304:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8305:  Function mwcRandomLongWord : LongWord
8306:  Function urnRandomLongWord : LongWord
8307:  Function moaRandomFloat : Extended
8308:  Function mwcRandomFloat : Extended
8309:  Function RandomNormalF : Extended
8310:  Procedure SelfTestCRandom
8311: end;
8312:
8313: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8314: begin
8315:  // PIntArray', '^TIntArray // will not work
8316:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8317:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8318:  Function synMax( x, y : integer) : integer
8319:  Function synMin( x, y : integer) : integer
8320:  Function synMinMax( x, mi, ma : integer) : integer
8321:  Procedure synSwapInt( var l, r : integer)
8322:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8323:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8324:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8325:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8326:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8327:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8328:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8329:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8330:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8331:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8332:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8333:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8334:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8335:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8336:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8337:  ('C3_NONSPACING','LongInt'( 1);
8338:  'C3_DIACRITIC','LongInt'( 2);
8339:  'C3_VOWELMARK','LongInt'( 4);
8340:  ('C3_SYMBOL','LongInt'( 8);
8341:  ('C3_KATAKANA','LongWord( $0010);
8342:  ('C3_HIRAGANA','LongWord( $0020);
8343:  ('C3_HALFWIDTH','LongWord( $0040);
8344:  ('C3_FULLWIDTH','LongWord( $0080);
8345:  ('C3_IDEOGRAPH','LongWord( $0100);
8346:  ('C3_KASHIDA','LongWord( $0200);
8347:  ('C3_LEXICAL','LongWord( $0400);
8348:  ('C3_ALPHA','LongWord( $8000);
8349:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8350:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8351:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8352:  Function synIsStringType( Value : Word) : TStringType
8353:  Function synGetEOL( Line : PChar) : PChar
8354:  Function synEncodeString( s : string) : string
8355:  Function synDecodeString( s : string) : string
8356:  Procedure synFreeAndNil( var Obj: TObject)
8357:  Procedure synAssert( Expr : Boolean)
8358:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8359:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8360:   TReplaceFlags', 'set of TReplaceFlag )
8361:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8362:  Function synGetRValue( RGBValue : TColor) : byte
8363:  Function synGetGValue( RGBValue : TColor) : byte
8364:  Function synGetBValue( RGBValue : TColor) : byte
8365:  Function synRGB( r, g, b : Byte) : Cardinal
8366: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8367:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8368:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8369:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8370:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8371: end;
8372:
8373:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8374:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8375:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8376:
8377: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8378: begin
8379:  Function STimeZoneBias : integer
```

```
8380:  Function TimeZone : string
8381:  Function Rfc822DateTime( t : TDateTime) : string
8382:  Function CDateTime( t : TDateTime) : string
8383:  Function SimpleDateTime( t : TDateTime) : string
8384:  Function AnsiCDateTime( t : TDateTime) : string
8385:  Function GetMonthNumber( Value : String) : integer
8386:  Function GetTimeFromStr( Value : string) : TDateTime
8387:  Function GetDateMDYFromStr( Value : string) : TDateTime
8388:  Function DecodeRfcDateTime( Value : string) : TDateTime
8389:  Function GetUTTime : TDateTime
8390:  Function SetUTTime( Newdt : TDateTime) : Boolean
8391:  Function SGetTick : LongWord
8392:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8393:  Function CodeInt( Value : Word) : Ansistring
8394:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8395:  Function CodeLongInt( Value : LongInt) : Ansistring
8396:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8397:  Function DumpStr( const Buffer : Ansistring) : string
8398:  Function DumpExStr( const Buffer : Ansistring) : string
8399:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8400:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8401:  Function TrimSPLeft( const S : string) : string
8402:  Function TrimSPRight( const S : string) : string
8403:  Function TrimSP( const S : string) : string
8404:  Function SeparateLeft( const Value, Delimiter : string) : string
8405:  Function SeparateRight( const Value, Delimiter : string) : string
8406:  Function SGetParameter( const Value, Parameter : string) : string
8407:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8408:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8409:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8410:  Function GetEmailAddr( const Value : string) : string
8411:  Function GetEmailDesc( Value : string) : string
8412:  Function CStrToHex( const Value : Ansistring) : string
8413:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8414:  Function CBinToInt( const Value : string) : Integer
8415:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8416:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8417:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8418:  Function CRPos( const Sub, Value : String) : Integer
8419:  Function FetchBin( var Value : string; const Delimiter : string) : string
8420:  Function CFetch( var Value : string; const Delimiter : string) : string
8421:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8422:  Function IsBinaryString( const Value : AnsiString) : Boolean
8423:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8424:  Procedure StringsTrim( const value : TStrings)
8425:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8426:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8427:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8428:  Function CCountOfChar( const Value : string; aChr : char) : integer
8429:  Function UnquoteStr( const Value : string; Quote : Char) : string
8430:  Function QuoteStr( const Value : string; Quote : Char) : string
8431:  Procedure HeadersToList( const Value : TStrings)
8432:  Procedure ListToHeaders( const Value : TStrings)
8433:  Function SwapBytes( Value : integer) : integer
8434:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8435:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8436:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8437:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8438:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8439:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8440: end;
8441:
8442: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8443: begin
8444:  ('CrcBufSize','LongInt'( 2048);
8445:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8446:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8447:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8448:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8449:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8450:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8451:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8452:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8453:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8454:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8455:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8456:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8457:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8458:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8459:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8460: end;
8461:
8462: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8463: begin
8464:  function CreateOleObject(const ClassName: String): IDispatch;
8465:  function GetActiveOleObject(const ClassName: String): IDispatch;
8466:  function ProgIDToClassID(const ProgID: string): TGUID;
8467:  function ClassIDToProgID(const ClassID: TGUID): string;
8468:  function CreateClassID: string;
```

```
8469:  function CreateGUIDString: string;
8470:  function CreateGUIDID: string;
8471:  procedure OleError(ErrorCode: longint)
8472:  procedure OleCheck(Result: HResult);
8473: end;
8474:
8475:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8476:  Function xGetActiveOleObject( const ClassName : string) : Variant
8477:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8478:  Function DllCanUnloadNow : HResult
8479:  Function DllRegisterServer : HResult
8480:  Function DllUnregisterServer : HResult
8481:  Function VarFromInterface( Unknown : IUnknown) : Variant
8482:  Function VarToInterface( const V : Variant) : IDispatch
8483:  Function VarToAutoObject( const V : Variant) : TAutoObject
8484:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8485:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8486:  Procedure OleError( ErrorCode : HResult)
8487:  Procedure OleCheck( Result : HResult)
8488:  Function StringToClassID( const S : string) : TCLSID
8489:  Function ClassIDToString( const ClassID : TCLSID) : string
8490:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8491:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8492:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8493:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8494:  Function xGUIDToString( const ClassID : TGUID) : string
8495:  Function xStringToGUID( const S : string) : TGUID
8496:  Function xGetModuleName( Module : HMODULE) : string
8497:  Function xAcquireExceptionObject : TObject
8498:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8499:  Function xUtf8Encode( const WS : WideString) : UTF8String
8500:  Function xUtf8Decode( const S : UTF8String) : WideString
8501:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8502:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8503:  Function XRTLHandleCOMException : HResult
8504:  Procedure XRTLCheckArgument( Flag : Boolean)
8505:  //Procedure XRTLCheckOutArgument( out Arg)
8506:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8507:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8508:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8509:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8510:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8511:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8512: function   XRTLDefaultCategoryManager: IUnknown;
8513: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8514: // ICatRegister helper functions
8515: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8516:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8517:                                        const CategoryManager: IUnknown = nil): HResult;
8518: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8519:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8520:                                        const CategoryManager: IUnknown = nil): HResult;
8521: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8522:                                        const CategoryManager: IUnknown = nil): HResult;
8523: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8524:                                          const CategoryManager: IUnknown = nil): HResult;
8525: // ICatInformation helper functions
8526: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8527:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8528:                                       const CategoryManager: IUnknown = nil): HResult;
8529: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8530:                                const CategoryManager: IUnknown = nil): HResult;
8531: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8532:                                     const CategoryManager: IUnknown = nil): HResult;
8533: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8534:                                       const CategoryManager: IUnknown = nil): HResult;
8535: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8536:                    const ADelete: Boolean = True): WideString;
8537: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8538: Function XRTLGetVariantAsString( const Value : Variant) : string
8539: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8540: Function XRTLGetTimeZones : TXRTLTimeZones
8541: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8542: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8543: Function GMTNow : TDateTime
8544: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8545: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8546: Procedure XRTLNotImplemented
8547: Procedure XRTLRaiseError( E : Exception)
8548: Procedure XRTLRaise( E : Exception)');
8549: Procedure XRaise( E : Exception)');
8550: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8551:
8552:
8553: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8554: begin
```

```
8555:   SIRegister_IXRTLValue(CL);
8556:   SIRegister_TXRTLValue(CL);
8557:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8558:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8559: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8560: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8561: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8562: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8563: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8564: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8565: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8566: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8567: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8568: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8569: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8570: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8571: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8572: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8573: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8574: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8575: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8576: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8577: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8578: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8579: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8580: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8581: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8582: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8583: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8584: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8585: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8586:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8587: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8588: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8589: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8590: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8591: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8592: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8593: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8594: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8595: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
     ADetachOwnership:Boolean):TObject;
8596: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8597: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8598: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8599: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8600: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8601: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8602: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8603: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8604: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8605: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8606: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8607: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8608: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8609: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8610: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8611: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8612: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8613: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8614: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8615: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8616: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8617: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8618: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8619: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8620: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8621: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8622: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8623: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8624: end;
8625:
8626: //****************************unit uPSI_GR32;****************************************
8627:
8628:  Function Color32( WinColor : TColor) : TColor32;
8629:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8630:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8631:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8632:  Function WinColor( Color32 : TColor32) : TColor
8633:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8634:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8635:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8636:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8637:  Function RedComponent( Color32 : TColor32) : Integer
8638:  Function GreenComponent( Color32 : TColor32) : Integer
8639:  Function BlueComponent( Color32 : TColor32) : Integer
8640:  Function AlphaComponent( Color32 : TColor32) : Integer
8641:  Function Intensity( Color32 : TColor32) : Integer
8642:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
```

```
8643:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8644:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8645:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8646:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8647:  Function WinPalette( const P : TPalette32) : HPALETTE
8648:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8649:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8650:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8651:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8652:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8653:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8654:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8655:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8656:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8657:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8658:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8659:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8660:  Function FixedRect1( const ARect : TRect) : TRect;
8661:  Function FixedRect2( const FR : TFloatRect) : TRect;
8662:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8663:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8664:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8665:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8666:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8667:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8668:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8669:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8670:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8671:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8672:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8673:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8674:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8675:  Function IsRectEmpty( const R : TRect) : Boolean;
8676:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8677:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8678:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8679:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8680:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8681:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8682:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8683:  Function MessageBeep( uType : UINT) : BOOL
8684:  Function ShowCursor( bShow : BOOL) : Integer
8685:  Function SetCursorPos( X, Y : Integer) : BOOL
8686:  Function SetCursor( hCursor : HICON) : HCURSOR
8687:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8688:  //Function ClipCursor( lpRect : PRect) : BOOL
8689:  Function GetClipCursor( var lpRect : TRect) : BOOL
8690:  Function GetCursor : HCURSOR
8691:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8692:  Function GetCaretBlinkTime : UINT
8693:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8694:  Function DestroyCaret : BOOL
8695:  Function HideCaret( hWnd : HWND) : BOOL
8696:  Function ShowCaret( hWnd : HWND) : BOOL
8697:  Function SetCaretPos( X, Y : Integer) : BOOL
8698:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8699:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8700:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8701:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8702:  Function WindowFromPoint( Point : TPoint) : HWND
8703:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8704:
8705:
8706: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8707: begin
8708:  Function FixedFloor( A : TFixed) : Integer
8709:  Function FixedCeil( A : TFixed) : Integer
8710:  Function FixedMul( A, B : TFixed) : TFixed
8711:  Function FixedDiv( A, B : TFixed) : TFixed
8712:  Function OneOver( Value : TFixed) : TFixed
8713:  Function FixedRound( A : TFixed) : Integer
8714:  Function FixedSqr( Value : TFixed) : TFixed
8715:  Function FixedSqrtLP( Value : TFixed) : TFixed
8716:  Function FixedSqrtHP( Value : TFixed) : TFixed
8717:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8718:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8719:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8720:  Function GRHypot( const X, Y : TFloat) : TFloat;
8721:  Function Hypot1( const X, Y : Integer) : Integer;
8722:  Function FastSqrt( const Value : TFloat) : TFloat
8723:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8724:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8725:  Function FastInvSqrt( const Value : Single) : Single;
8726:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8727:  Function GRIsPowerOf2( Value : Integer) : Boolean
8728:  Function PrevPowerOf2( Value : Integer) : Integer
8729:  Function NextPowerOf2( Value : Integer) : Integer
8730:  Function Average( A, B : Integer) : Integer
8731:  Function GRSign( Value : Integer) : Integer
```

```
8732:  Function FloatMod( x, y : Double) : Double
8733: end;
8734:
8735: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8736: begin
8737:  Function Clamp( const Value : Integer) : Integer;
8738:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8739:  Function StackAlloc( Size : Integer) : Pointer
8740:  Procedure StackFree( P : Pointer)
8741:  Procedure Swap( var A, B : Pointer);
8742:  Procedure Swap1( var A, B : Integer);
8743:  Procedure Swap2( var A, B : TFixed);
8744:  Procedure Swap3( var A, B : TColor32);
8745:  Procedure TestSwap( var A, B : Integer);
8746:  Procedure TestSwap1( var A, B : TFixed);
8747:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8748:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8749:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8750:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8751:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8752:  Function GRMin( const A, B, C : Integer) : Integer;
8753:  Function GRMax( const A, B, C : Integer) : Integer;
8754:  Function Clamp( Value, Max : Integer) : Integer;
8755:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8756:  Function Wrap( Value, Max : Integer) : Integer;
8757:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8758:  Function Wrap3( Value, Max : Single) : Single;;
8759:  Function WrapPow2( Value, Max : Integer) : Integer;
8760:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8761:  Function Mirror( Value, Max : Integer) : Integer;
8762:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8763:  Function MirrorPow2( Value, Max : Integer) : Integer;
8764:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8765:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8766:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8767:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8768:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8769:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8770:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8771:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8772:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8773:  Function Div255( Value : Cardinal) : Cardinal
8774:  Function SAR_4( Value : Integer) : Integer
8775:  Function SAR_8( Value : Integer) : Integer
8776:  Function SAR_9( Value : Integer) : Integer
8777:  Function SAR_11( Value : Integer) : Integer
8778:  Function SAR_12( Value : Integer) : Integer
8779:  Function SAR_13( Value : Integer) : Integer
8780:  Function SAR_14( Value : Integer) : Integer
8781:  Function SAR_15( Value : Integer) : Integer
8782:  Function SAR_16( Value : Integer) : Integer
8783:  Function ColorSwap( WinColor : TColor) : TColor32
8784: end;
8785:
8786: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8787: begin
8788:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8789:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8790:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8791:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8792:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8793:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8794:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8795:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8796:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8797:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8798:  Function CreateBitmask( Components : TColor32Components) : TColor32
8799:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8800:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8801:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8802: end;
8803:
8804:
8805: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8806: begin
8807:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8808:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8809:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8810:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8811:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8812:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8813:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8814:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8815:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8816:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8817:    //+'tedRangeBuffer; MoreData : Boolean; end
```

```
8818:  Function NtfsSetSparse( const FileName : string) : Boolean
8819:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8820:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8821:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
       Ranges:TNtfsAllocRanges):Boolean;
8822:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
       Index:Integer):TFileAllocatedRangeBuffer
8823:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8824:  Function NtfsGetSparse( const FileName : string) : Boolean
8825:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8826:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8827:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8828:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8829:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8830:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8831:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8832:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8833:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8834:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8835:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8836:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8837:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8838:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8839:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8840:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8841:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8842:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8843:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8844:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8845:   AddTypeS('TStreamIds', 'set of TStreamId
8846:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8847:    +': ___Pointer; StreamIds : TStreamIds; end
8848:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8849:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8850:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8851:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8852:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8853:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8854:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8855:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8856:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
       List:TStrings):Bool;
8857:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8858:  Function JclAppInstances : TJclAppInstances;
8859:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8860:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8861:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8862:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8863:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8864:
8865:
8866: (*----------------------------------------------------------------------------*)
8867: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8868: begin
8869:   FindClass('TOBJECT'),'EJclGraphicsError
8870:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8871:   TDynPointArray', 'array of TPoint
8872:   TDynDynPointArrayArray', 'array of TDynPointArray
8873:   TPointF', 'record X : Single; Y : Single; end
8874:   TDynPointArrayF', 'array of TPointF
8875:   TDrawMode2', '( dmOpaque, dmBlend )
8876:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8877:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8878:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8879:   TMatrix3d', 'record array[0..2,0..2] of extended end
8880:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8881:   TScanLine', 'array of Integer
8882:   TScanLines', 'array of TScanLine
8883:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8884:   TGradientDirection', '( gdVertical, gdHorizontal )
8885:   TPolyFillMode', '( fmAlternate, fmWinding )
8886:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8887:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8888:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8889:   SIRegister_TJclDesktopCanvas(CL);
8890:   FindClass('TOBJECT'),'TJclRegion
8891:   SIRegister_TJclRegionInfo(CL);
8892:   SIRegister_TJclRegion(CL);
8893:   SIRegister_TJclThreadPersistent(CL);
8894:   SIRegister_TJclCustomMap(CL);
8895:   SIRegister_TJclBitmap32(CL);
8896:   SIRegister_TJclByteMap(CL);
8897:   SIRegister_TJclTransformation(CL);
8898:   SIRegister_TJclLinearTransformation(CL);
8899:  Procedure Stretch(NewWidth,
       NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8900:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8901:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8902:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
```

```
8903:  Procedure BitmapToJPeg( const FileName : string)
8904:  Procedure JPegToBitmap( const FileName : string)
8905:  Function ExtractIconCount( const FileName : string) : Integer
8906:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8907:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8908:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8909:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8910:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8911:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8912:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8913:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8914:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8915:  Procedure ScreenShot1( bm : TBitmap);
8916:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8917:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8918:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8919:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8920:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8921:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8922:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8923:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8924:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8925:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8926:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8927:  Procedure Invert( Dst, Src : TJclBitmap32)
8928:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8929:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8930:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8931:  Procedure SetGamma( Gamma : Single)
8932: end;
8933:
8934: (*----------------------------------------------------------------------------*)
8935: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8936: begin
8937:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8938:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8939:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8940:  Function LockedDec( var Target : Integer) : Integer
8941:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8942:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8943:  Function LockedExchangeDec( var Target : Integer) : Integer
8944:  Function LockedExchangeInc( var Target : Integer) : Integer
8945:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8946:  Function LockedInc( var Target : Integer) : Integer
8947:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8948:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8949:   SIRegister_TJclDispatcherObject(CL);
8950:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8951:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8952:   SIRegister_TJclCriticalSection(CL);
8953:   SIRegister_TJclCriticalSectionEx(CL);
8954:   SIRegister_TJclEvent(CL);
8955:   SIRegister_TJclWaitableTimer(CL);
8956:   SIRegister_TJclSemaphore(CL);
8957:   SIRegister_TJclMutex(CL);
8958:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8959:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8960:   SIRegister_TJclOptex(CL);
8961:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8962:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8963:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8964:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8965:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8966:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8967:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8968:   PMeteredSection', '^TMeteredSection // will not work
8969:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8970:   SIRegister_TJclMeteredSection(CL);
8971:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8972:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8973:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8974:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8975:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8976:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8977:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8978:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8979:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8980:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8981:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8982:   FindClass('TOBJECT'),'EJclCriticalSectionError
8983:   FindClass('TOBJECT'),'EJclEventError
8984:   FindClass('TOBJECT'),'EJclWaitableTimerError
8985:   FindClass('TOBJECT'),'EJclSemaphoreError
```

```
8986:   FindClass('TOBJECT'),'EJclMutexError
8987:   FindClass('TOBJECT'),'EJclMeteredSectionError
8988: end;
8989:
8990:
8991: //*************************unit uPSI_mORMotReport;
8992: Procedure SetCurrentPrinterAsDefault
8993: Function CurrentPrinterName : string
8994: Function mCurrentPrinterPaperSize : string
8995: Procedure UseDefaultPrinter
8996:
8997: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
8998: begin
8999:   with FindClass('TOBJECT'), 'TStream') do begin
9000:     IsAbstract := True;
9001:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
9002:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
9003:     function Read(Buffer:String;Count:LongInt):LongInt
9004:     function Write(Buffer:String;Count:LongInt):LongInt
9005:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
9006:     function WriteString(Buffer:String;Count:LongInt):LongInt
9007:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
9008:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
9009:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
9010:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
9011:
9012:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
9013:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
9014:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
9015:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
9016:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
9017:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
9018:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
9019:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
9020:
9021:     function Seek(Offset:LongInt;Origin:Word):LongInt
9022:     procedure ReadBuffer(Buffer:String;Count:LongInt)
9023:     procedure WriteBuffer(Buffer:String;Count:LongInt)
9024:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
9025:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
9026:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
9027:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
9028:
9029:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9030:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9031:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9032:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9033:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9034:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9035:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9036:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9037:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9038:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9039:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9040:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9041:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9042:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9043:
9044:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9045:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9046:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
9047:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
9048:     //READBUFFERAC
9049:     function InstanceSize: Longint
9050:     Procedure FixupResourceHeader( FixupInfo : Integer)
9051:     Procedure ReadResHeader
9052:
9053:     {$IFDEF DELPHI4UP}
9054:     function CopyFrom(Source:TStream;Count:Int64):LongInt
9055:     {$ELSE}
9056:     function CopyFrom(Source:TStream;Count:Integer):LongInt
9057:     {$ENDIF}
9058:     RegisterProperty('Position', 'LongInt', iptrw);
9059:     RegisterProperty('Size', 'LongInt', iptrw);
9060:   end;
9061: end;
9062:
9063:
9064: { ***************************************************************
9065:   Unit DMATH - Interface for DMATH.DLL
9066:    *************************************************************** }
9067: // see more docs/dmath_manual.pdf
9068:
9069: Function InitEval : Integer
9070: Procedure SetVariable( VarName : Char; Value : Float)
9071: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9072: Function Eval( ExpressionString : String) : Float
9073:
9074: unit dmath; //types are in built, others are external in DLL
```

```
9075: interface
9076: {$IFDEF DELPHI}
9077: uses
9078:   StdCtrls, Graphics;
9079: {$ENDIF}
9080: { ------------------------------------------------------------------
9081:   Types and constants
9082:   ------------------------------------------------------------------ }
9083: {$i types.inc}
9084: { ------------------------------------------------------------------
9085:   Error handling
9086:   ------------------------------------------------------------------ }
9087: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9088: { Sets the error code }
9089: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9090: { Sets error code and default function value }
9091: function MathErr : Integer; external 'dmath';
9092: { Returns the error code }
9093: { ------------------------------------------------------------------
9094:   Dynamic arrays
9095:   ------------------------------------------------------------------ }
9096: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9097: { Sets the auto-initialization of arrays }
9098: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9099: { Creates floating point vector V[0..Ub] }
9100: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9101: { Creates integer vector V[0..Ub] }
9102: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9103: { Creates complex vector V[0..Ub] }
9104: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9105: { Creates boolean vector V[0..Ub] }
9106: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9107: { Creates string vector V[0..Ub] }
9108: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9109: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9110: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9111: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9112: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9113: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9114: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9115: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9116: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9117: { Creates string matrix A[0..Ub1, 0..Ub2] }
9118: { ------------------------------------------------------------------
9119:   Minimum, maximum, sign and exchange
9120:   ------------------------------------------------------------------ }
9121: function FMin(X, Y : Float) : Float; external 'dmath';
9122: { Minimum of 2 reals }
9123: function FMax(X, Y : Float) : Float; external 'dmath';
9124: { Maximum of 2 reals }
9125: function IMin(X, Y : Integer) : Integer; external 'dmath';
9126: { Minimum of 2 integers }
9127: function IMax(X, Y : Integer) : Integer; external 'dmath';
9128: { Maximum of 2 integers }
9129: function Sgn(X : Float) : Integer; external 'dmath';
9130: { Sign (returns 1 if X = 0) }
9131: function Sgn0(X : Float) : Integer; external 'dmath';
9132: { Sign (returns 0 if X = 0) }
9133: function DSgn(A, B : Float) : Float; external 'dmath';
9134: { Sgn(B) * |A| }
9135: procedure FSwap(var X, Y : Float); external 'dmath';
9136: { Exchange 2 reals }
9137: procedure ISwap(var X, Y : Integer); external 'dmath';
9138: { Exchange 2 integers }
9139: { ------------------------------------------------------------------
9140:   Rounding functions
9141:   ------------------------------------------------------------------ }
9142: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9143: { Rounds X to N decimal places }
9144: function Ceil(X : Float) : Integer; external 'dmath';
9145: { Ceiling function }
9146: function Floor(X : Float) : Integer; external 'dmath';
9147: { Floor function }
9148: { ------------------------------------------------------------------
9149:   Logarithms, exponentials and power
9150:   ------------------------------------------------------------------ }
9151: function Expo(X : Float) : Float; external 'dmath';
9152: { Exponential }
9153: function Exp2(X : Float) : Float; external 'dmath';
9154: { 2^X }
9155: function Exp10(X : Float) : Float; external 'dmath';
9156: { 10^X }
9157: function Log(X : Float) : Float; external 'dmath';
9158: { Natural log }
9159: function Log2(X : Float) : Float; external 'dmath';
9160: { Log, base 2 }
9161: function Log10(X : Float) : Float; external 'dmath';
9162: { Decimal log }
9163: function LogA(X, A : Float) : Float; external 'dmath';
```

```
9164: { Log, base A }
9165: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9166: { X^N }
9167: function Power(X, Y : Float) : Float; external 'dmath';
9168: { X^Y, X >= 0 }
9169: { ---------------------------------------------------------------
9170:   Trigonometric functions
9171:   --------------------------------------------------------------- }
9172: function Pythag(X, Y : Float) : Float; external 'dmath';
9173: { Sqrt(X^2 + Y^2) }
9174: function FixAngle(Theta : Float) : Float; external 'dmath';
9175: { Set Theta in -Pi..Pi }
9176: function Tan(X : Float) : Float; external 'dmath';
9177: { Tangent }
9178: function ArcSin(X : Float) : Float; external 'dmath';
9179: { Arc sinus }
9180: function ArcCos(X : Float) : Float; external 'dmath';
9181: { Arc cosinus }
9182: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9183: { Angle (Ox, OM) with M(X,Y) }
9184: { ---------------------------------------------------------------
9185:   Hyperbolic functions
9186:   --------------------------------------------------------------- }
9187: function Sinh(X : Float) : Float; external 'dmath';
9188: { Hyperbolic sine }
9189: function Cosh(X : Float) : Float; external 'dmath';
9190: { Hyperbolic cosine }
9191: function Tanh(X : Float) : Float; external 'dmath';
9192: { Hyperbolic tangent }
9193: function ArcSinh(X : Float) : Float; external 'dmath';
9194: { Inverse hyperbolic sine }
9195: function ArcCosh(X : Float) : Float; external 'dmath';
9196: { Inverse hyperbolic cosine }
9197: function ArcTanh(X : Float) : Float; external 'dmath';
9198: { Inverse hyperbolic tangent }
9199: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9200: { Sinh & Cosh }
9201: { ---------------------------------------------------------------
9202:   Gamma function and related functions
9203:   --------------------------------------------------------------- }
9204: function Fact(N : Integer) : Float; external 'dmath';
9205: { Factorial }
9206: function SgnGamma(X : Float) : Integer; external 'dmath';
9207: { Sign of Gamma function }
9208: function Gamma(X : Float) : Float; external 'dmath';
9209: { Gamma function }
9210: function LnGamma(X : Float) : Float; external 'dmath';
9211: { Logarithm of Gamma function }
9212: function Stirling(X : Float) : Float; external 'dmath';
9213: { Stirling's formula for the Gamma function }
9214: function StirLog(X : Float) : Float; external 'dmath';
9215: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9216: function DiGamma(X : Float ) : Float; external 'dmath';
9217: { Digamma function }
9218: function TriGamma(X : Float ) : Float; external 'dmath';
9219: { Trigamma function }
9220: function IGamma(A, X : Float) : Float; external 'dmath';
9221: { Incomplete Gamma function}
9222: function JGamma(A, X : Float) : Float; external 'dmath';
9223: { Complement of incomplete Gamma function }
9224: function InvGamma(A, P : Float) : Float; external 'dmath';
9225: { Inverse of incomplete Gamma function }
9226: function Erf(X : Float) : Float; external 'dmath';
9227: { Error function }
9228: function Erfc(X : Float) : Float; external 'dmath';
9229: { Complement of error function }
9230: { ---------------------------------------------------------------
9231:   Beta function and related functions
9232:   --------------------------------------------------------------- }
9233: function Beta(X, Y : Float) : Float; external 'dmath';
9234: { Beta function }
9235: function IBeta(A, B, X : Float) : Float; external 'dmath';
9236: { Incomplete Beta function }
9237: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9238: { Inverse of incomplete Beta function }
9239: { ---------------------------------------------------------------
9240:   Lambert's function
9241:   --------------------------------------------------------------- }
9242: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9243:  ---------------------------------------------------------------
9244:   Binomial distribution
9245:   --------------------------------------------------------------- }
9246: function Binomial(N, K : Integer) : Float; external 'dmath';
9247: { Binomial coefficient C(N,K) }
9248: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9249: { Probability of binomial distribution }
9250: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9251: { Cumulative probability for binomial distrib. }
9252: { ---------------------------------------------------------------
```

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

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

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

```
9520:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9521: { Evaluates a rational fraction }
9522: function RootPol1(A, B  : Float;
9523:                   var X : Float) : Integer; external 'dmath';
9524: { Solves the linear equation A + B * X = 0 }
9525: function RootPol2(Coef : TVector;
9526:                   Z    : TCompVector) : Integer; external 'dmath';
9527: { Solves a quadratic equation }
9528: function RootPol3(Coef : TVector;
9529:                   Z    : TCompVector) : Integer; external 'dmath';
9530: { Solves a cubic equation }
9531: function RootPol4(Coef : TVector;
9532:                   Z    : TCompVector) : Integer; external 'dmath';
9533: { Solves a quartic equation }
9534: function RootPol(Coef : TVector;
9535:                  Deg  : Integer;
9536:                  Z    : TCompVector) : Integer; external 'dmath';
9537: { Solves a polynomial equation }
9538: function SetRealRoots(Deg : Integer;
9539:                       Z   : TCompVector;
9540:                       Tol : Float) : Integer; external 'dmath';
9541: { Set the imaginary part of a root to zero }
9542: procedure SortRoots(Deg : Integer;
9543:                     Z   : TCompVector); external 'dmath';
9544: { Sorts the roots of a polynomial }
9545: { ------------------------------------------------------------------
9546:   Numerical integration and differential equations
9547:   ------------------------------------------------------------------ }
9548: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9549: { Integration by trapezoidal rule }
9550: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9551: { Integral from A to B }
9552: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9553: { Integral from 0 to B }
9554: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9555: { Convolution product at time T }
9556: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9557: { Convolution by trapezoidal rule }
9558: procedure RKF45(F                   : TDiffEqs;
9559:                 Neqn                : Integer;
9560:                 Y, Yp               : TVector;
9561:                 var T               : Float;
9562:                 Tout, RelErr, AbsErr : Float;
9563:                 var Flag            : Integer); external 'dmath';
9564: { Integration of a system of differential equations }
9565: { ------------------------------------------------------------------
9566:   Fast Fourier Transform
9567:   ------------------------------------------------------------------ }
9568: procedure FFT(NumSamples        : Integer;
9569:               InArray, OutArray : TCompVector); external 'dmath';
9570: { Fast Fourier Transform }
9571: procedure IFFT(NumSamples        : Integer;
9572:                InArray, OutArray : TCompVector); external 'dmath';
9573: { Inverse Fast Fourier Transform }
9574: procedure FFT_Integer(NumSamples     : Integer;
9575:                       RealIn, ImagIn : TIntVector;
9576:                       OutArray       : TCompVector); external 'dmath';
9577: { Fast Fourier Transform for integer data }
9578: procedure FFT_Integer_Cleanup; external 'dmath';
9579: { Clear memory after a call to FFT_Integer }
9580: procedure CalcFrequency(NumSamples,
9581:                         FrequencyIndex : Integer;
9582:                         InArray        : TCompVector;
9583:                         var FFT        : Complex); external 'dmath';
9584: { Direct computation of Fourier transform }
9585: { ------------------------------------------------------------------
9586:   Random numbers
9587:   ------------------------------------------------------------------ }
9588: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9589: { Select generator }
9590: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9591: { Initialize generator }
9592: function IRanGen : RNG_IntType; external 'dmath';
9593: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9594: function IRanGen31 : RNG_IntType; external 'dmath';
9595: { 31-bit random integer in [0 .. 2^31 - 1] }
9596: function RanGen1 : Float; external 'dmath';
9597: { 32-bit random real in [0,1] }
9598: function RanGen2 : Float; external 'dmath';
9599: { 32-bit random real in [0,1) }
9600: function RanGen3 : Float; external 'dmath';
9601: { 32-bit random real in (0,1) }
9602: function RanGen53 : Float; external 'dmath';
9603: { 53-bit random real in [0,1) }
9604: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9605: { Initializes the 'Multiply with carry' random number generator }
9606: function IRanMWC : RNG_IntType; external 'dmath';
9607: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9608: procedure InitMT(Seed : RNG_IntType); external 'dmath';
```

```
9609: { Initializes Mersenne Twister generator with a seed }
9610: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9611:                         KeyLength : Word); external 'dmath';
9612: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9613: function IRanMT : RNG_IntType; external 'dmath';
9614: { Random integer from MT generator }
9615: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9616: { Initializes the UVAG generator with a string }
9617: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9618: { Initializes the UVAG generator with an integer }
9619: function IRanUVAG : RNG_IntType; external 'dmath';
9620: { Random integer from UVAG generator }
9621: function RanGaussStd : Float; external 'dmath';
9622: { Random number from standard normal distribution }
9623: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9624: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9625: procedure RanMult(M      : TVector; L      : TMatrix;
9626:                   Lb, Ub : Integer;
9627:                   X      : TVector); external 'dmath';
9628: { Random vector from multinormal distribution (correlated) }
9629: procedure RanMultIndep(M, S   : TVector;
9630:                        Lb, Ub : Integer;
9631:                        X      : TVector); external 'dmath';
9632: { Random vector from multinormal distribution (uncorrelated) }
9633: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9634: { Initializes Metropolis-Hastings parameters }
9635: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9636: { Returns Metropolis-Hastings parameters }
9637: procedure Hastings(Func      : TFuncNVar;
9638:                    T         : Float;
9639:                    X         : TVector;
9640:                    V         : TMatrix;
9641:                    Lb, Ub    : Integer;
9642:                    Xmat      : TMatrix;
9643:                    X_min     : TVector;
9644:                    var F_min : Float); external 'dmath';
9645: { Simulation of a probability density function by Metropolis-Hastings }
9646: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9647: { Initializes Simulated Annealing parameters }
9648: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9649: { Initializes log file }
9650: procedure SimAnn(Func          : TFuncNVar;
9651:                  X, Xmin, Xmax : TVector;
9652:                  Lb, Ub        : Integer;
9653:                  var F_min     : Float); external 'dmath';
9654: { Minimization of a function of several var. by simulated annealing }
9655: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9656: { Initializes Genetic Algorithm parameters }
9657: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9658: { Initializes log file }
9659: procedure GenAlg(Func          : TFuncNVar;
9660:                  X, Xmin, Xmax : TVector;
9661:                  Lb, Ub        : Integer;
9662:                  var F_min     : Float); external 'dmath';
9663: { Minimization of a function of several var. by genetic algorithm }
9664: { ---------------------------------------------------------------
9665:   Statistics
9666:   --------------------------------------------------------------- }
9667: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9668: { Mean of sample X }
9669: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9670: { Minimum of sample X }
9671: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9672: { Maximum of sample X }
9673: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9674: { Median of sample X }
9675: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9676: { Standard deviation estimated from sample X }
9677: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9678: { Standard deviation of population }
9679: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9680: { Correlation coefficient }
9681: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9682: { Skewness of sample X }
9683: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9684: { Kurtosis of sample X }
9685: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9686: { Quick sort (ascending order) }
9687: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9688: { Quick sort (descending order) }
9689: procedure Interval(X1, X2             : Float;
9690:                    MinDiv, MaxDiv     : Integer;
9691:                    var Min, Max, Step : Float); external 'dmath';
9692: { Determines an interval for a set of values }
9693: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9694:                     var XMin, XMax, XStep : Float); external 'dmath';
9695: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9696: procedure StudIndep(N1, N2         : Integer;
9697:                     M1, M2, S1, S2 : Float;
```

```
9698:                    var T          : Float;
9699:                    var DoF        : Integer); external 'dmath';
9700: { Student t-test for independent samples }
9701: procedure StudPaired(X, Y    : TVector;
9702:                      Lb, Ub  : Integer;
9703:                      var T   : Float;
9704:                      var DoF : Integer); external 'dmath';
9705: { Student t-test for paired samples }
9706: procedure AnOVa1(Ns               : Integer;
9707:                  N                : TIntVector;
9708:                  M, S             : TVector;
9709:                  var V_f, V_r, F  : Float;
9710:                  var DoF_f, DoF_r : Integer); external 'dmath';
9711: { One-way analysis of variance }
9712: procedure AnOVa2(NA, NB, Nobs : Integer;
9713:                  M, S         : TMatrix;
9714:                  V, F         : TVector;
9715:                  DoF          : TIntVector); external 'dmath';
9716: { Two-way analysis of variance }
9717: procedure Snedecor(N1, N2          : Integer;
9718:                    S1, S2          : Float;
9719:                    var F           : Float;
9720:                    var DoF1, DoF2 : Integer); external 'dmath';
9721: { Snedecor's F-test (comparison of two variances) }
9722: procedure Bartlett(Ns       : Integer;
9723:                    N        : TIntVector;
9724:                    S        : TVector;
9725:                    var Khi2 : Float;
9726:                    var DoF  : Integer); external 'dmath';
9727: { Bartlett's test (comparison of several variances) }
9728: procedure Mann_Whitney(N1, N2     : Integer;
9729:                        X1, X2     : TVector;
9730:                        var U, Eps : Float); external 'dmath';
9731: { Mann-Whitney test}
9732: procedure Wilcoxon(X, Y       : TVector;
9733:                    Lb, Ub     : Integer;
9734:                    var Ndiff  : Integer;
9735:                    var T, Eps : Float); external 'dmath';
9736: { Wilcoxon test }
9737: procedure Kruskal_Wallis(Ns      : Integer;
9738:                          N       : TIntVector;
9739:                          X       : TMatrix;
9740:                          var H   : Float;
9741:                          var DoF : Integer); external 'dmath';
9742: { Kruskal-Wallis test }
9743: procedure Khi2_Conform(N_cls    : Integer;
9744:                        N_estim  : Integer;
9745:                        Obs      : TIntVector;
9746:                        Calc     : TVector;
9747:                        var Khi2 : Float;
9748:                        var DoF  : Integer); external 'dmath';
9749: { Khi-2 test for conformity }
9750: procedure Khi2_Indep(N_lin    : Integer;
9751:                      N_col    : Integer;
9752:                      Obs      : TIntMatrix;
9753:                      var Khi2 : Float;
9754:                      var DoF  : Integer); external 'dmath';
9755: { Khi-2 test for independence }
9756: procedure Woolf_Conform(N_cls   : Integer;
9757:                         N_estim : Integer;
9758:                         Obs     : TIntVector;
9759:                         Calc    : TVector;
9760:                         var G   : Float;
9761:                         var DoF : Integer); external 'dmath';
9762: { Woolf's test for conformity }
9763: procedure Woolf_Indep(N_lin   : Integer;
9764:                       N_col   : Integer;
9765:                       Obs     : TIntMatrix;
9766:                       var G   : Float;
9767:                       var DoF : Integer); external 'dmath';
9768: { Woolf's test for independence }
9769: procedure DimStatClassVector(var C : TStatClassVector;
9770:                              Ub    : Integer); external 'dmath';
9771: { Allocates an array of statistical classes: C[0..Ub] }
9772: procedure Distrib(X       : TVector;
9773:                   Lb, Ub  : Integer;
9774:                   A, B, H : Float;
9775:                   C       : TStatClassVector); external 'dmath';
9776: { Distributes an array X[Lb..Ub] into statistical classes }
9777: { ----------------------------------------------------------------
9778:   Linear / polynomial regression
9779:   ---------------------------------------------------------------- }
9780: procedure LinFit(X, Y   : TVector;
9781:                  Lb, Ub : Integer;
9782:                  B      : TVector;
9783:                  V      : TMatrix); external 'dmath';
9784: { Linear regression : Y = B(0) + B(1) * X }
9785: procedure WLinFit(X, Y, S : TVector;
9786:                   Lb, Ub  : Integer;
```

```
9787:                   B       : TVector;
9788:                   V       : TMatrix); external 'dmath';
9789: { Weighted linear regression : Y = B(0) + B(1) * X }
9790: procedure SVDLinFit(X, Y   : TVector;
9791:                     Lb, Ub : Integer;
9792:                     SVDTol : Float;
9793:                     B      : TVector;
9794:                     V      : TMatrix); external 'dmath';
9795: { Unweighted linear regression by singular value decomposition }
9796: procedure WSVDLinFit(X, Y, S : TVector;
9797:                      Lb, Ub  : Integer;
9798:                      SVDTol  : Float;
9799:                      B       : TVector;
9800:                      V       : TMatrix); external 'dmath';
9801: { Weighted linear regression by singular value decomposition }
9802: procedure MulFit(X            : TMatrix;
9803:                  Y            : TVector;
9804:                  Lb, Ub, Nvar : Integer;
9805:                  ConsTerm     : Boolean;
9806:                  B            : TVector;
9807:                  V            : TMatrix); external 'dmath';
9808: { Multiple linear regression by Gauss-Jordan method }
9809: procedure WMulFit(X            : TMatrix;
9810:                   Y, S         : TVector;
9811:                   Lb, Ub, Nvar : Integer;
9812:                   ConsTerm     : Boolean;
9813:                   B            : TVector;
9814:                   V            : TMatrix); external 'dmath';
9815: { Weighted multiple linear regression by Gauss-Jordan method }
9816: procedure SVDFit(X            : TMatrix;
9817:                  Y            : TVector;
9818:                  Lb, Ub, Nvar : Integer;
9819:                  ConsTerm     : Boolean;
9820:                  SVDTol       : Float;
9821:                  B            : TVector;
9822:                  V            : TMatrix); external 'dmath';
9823: { Multiple linear regression by singular value decomposition }
9824: procedure WSVDFit(X            : TMatrix;
9825:                   Y, S         : TVector;
9826:                   Lb, Ub, Nvar : Integer;
9827:                   ConsTerm     : Boolean;
9828:                   SVDTol       : Float;
9829:                   B            : TVector;
9830:                   V            : TMatrix); external 'dmath';
9831: { Weighted multiple linear regression by singular value decomposition }
9832: procedure PolFit(X, Y        : TVector;
9833:                  Lb, Ub, Deg : Integer;
9834:                  B           : TVector;
9835:                  V           : TMatrix); external 'dmath';
9836: { Polynomial regression by Gauss-Jordan method }
9837: procedure WPolFit(X, Y, S     : TVector;
9838:                   Lb, Ub, Deg : Integer;
9839:                   B           : TVector;
9840:                   V           : TMatrix); external 'dmath';
9841: { Weighted polynomial regression by Gauss-Jordan method }
9842: procedure SVDPolFit(X, Y        : TVector;
9843:                     Lb, Ub, Deg : Integer;
9844:                     SVDTol      : Float;
9845:                     B           : TVector;
9846:                     V           : TMatrix); external 'dmath';
9847: { Unweighted polynomial regression by singular value decomposition }
9848: procedure WSVDPolFit(X, Y, S     : TVector;
9849:                      Lb, Ub, Deg : Integer;
9850:                      SVDTol      : Float;
9851:                      B           : TVector;
9852:                      V           : TMatrix); external 'dmath';
9853: { Weighted polynomial regression by singular value decomposition }
9854: procedure RegTest(Y, Ycalc : TVector;
9855:                   LbY, UbY : Integer;
9856:                   V        : TMatrix;
9857:                   LbV, UbV : Integer;
9858:                   var Test : TRegTest); external 'dmath';
9859: { Test of unweighted regression }
9860: procedure WRegTest(Y, Ycalc, S : TVector;
9861:                    LbY, UbY    : Integer;
9862:                    V           : TMatrix;
9863:                    LbV, UbV    : Integer;
9864:                    var Test    : TRegTest); external 'dmath';
9865: { Test of weighted regression }
9866: { ---------------------------------------------------------------
9867:   Nonlinear regression
9868:   --------------------------------------------------------------- }
9869: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9870: { Sets the optimization algorithm for nonlinear regression }
9871: function GetOptAlgo : TOptAlgo; external 'dmath';
9872: { Returns the optimization algorithm }
9873: procedure SetMaxParam(N : Byte); external 'dmath';
9874: { Sets the maximum number of regression parameters for nonlinear regression }
9875: function GetMaxParam : Byte; external 'dmath';
```

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

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

```
10054:                   MaxIter  : Integer;
10055:                   Tol      : Float;
10056:                   B        : TVector;
10057:                   V        : TMatrix); external 'dmath';
10058: { Unweighted fit of logistic function }
10059: procedure WLogiFit(X, Y, S  : TVector;
10060:                    Lb, Ub   : Integer;
10061:                    ConsTerm : Boolean;
10062:                    General  : Boolean;
10063:                    MaxIter  : Integer;
10064:                    Tol      : Float;
10065:                    B        : TVector;
10066:                    V        : TMatrix); external 'dmath';
10067: { Weighted fit of logistic function }
10068: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10069: { Returns the value of the logistic function at point X }
10070: procedure PKFit(X, Y    : TVector;
10071:                 Lb, Ub  : Integer;
10072:                 MaxIter : Integer;
10073:                 Tol     : Float;
10074:                 B       : TVector;
10075:                 V       : TMatrix); external 'dmath';
10076: { Unweighted fit of the acid-base titration curve }
10077: procedure WPKFit(X, Y, S : TVector;
10078:                  Lb, Ub  : Integer;
10079:                  MaxIter : Integer;
10080:                  Tol     : Float;
10081:                  B       : TVector;
10082:                  V       : TMatrix); external 'dmath';
10083: { Weighted fit of the acid-base titration curve }
10084: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10085: { Returns the value of the acid-base titration function at point X }
10086: procedure PowFit(X, Y    : TVector;
10087:                  Lb, Ub  : Integer;
10088:                  MaxIter : Integer;
10089:                  Tol     : Float;
10090:                  B       : TVector;
10091:                  V       : TMatrix); external 'dmath';
10092: { Unweighted fit of power function }
10093: procedure WPowFit(X, Y, S : TVector;
10094:                   Lb, Ub  : Integer;
10095:                   MaxIter : Integer;
10096:                   Tol     : Float;
10097:                   B       : TVector;
10098:                   V       : TMatrix); external 'dmath';
10099: { Weighted fit of power function }
10100:
10101: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10102: { Returns the value of the power function at point X }
10103: procedure GammaFit(X, Y    : TVector;
10104:                    Lb, Ub  : Integer;
10105:                    MaxIter : Integer;
10106:                    Tol     : Float;
10107:                    B       : TVector;
10108:                    V       : TMatrix); external 'dmath';
10109: { Unweighted fit of gamma distribution function }
10110: procedure WGammaFit(X, Y, S : TVector;
10111:                     Lb, Ub  : Integer;
10112:                     MaxIter : Integer;
10113:                     Tol     : Float;
10114:                     B       : TVector;
10115:                     V       : TMatrix); external 'dmath';
10116: { Weighted fit of gamma distribution function }
10117: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10118: { Returns the value of the gamma distribution function at point X }
10119: { ---------------------------------------------------------------
10120:   Principal component analysis
10121:   --------------------------------------------------------------- }
10122: procedure VecMean(X            : TMatrix;
10123:                   Lb, Ub, Nvar : Integer;
10124:                   M            : TVector); external 'dmath';
10125: { Computes the mean vector M from matrix X }
10126: procedure VecSD(X            : TMatrix;
10127:                 Lb, Ub, Nvar : Integer;
10128:                 M, S         : TVector); external 'dmath';
10129: { Computes the vector of standard deviations S from matrix X }
10130: procedure MatVarCov(X            : TMatrix;
10131:                     Lb, Ub, Nvar : Integer;
10132:                     M            : TVector;
10133:                     V            : TMatrix); external 'dmath';
10134: { Computes the variance-covariance matrix V from matrix X }
10135: procedure MatCorrel(V    : TMatrix;
10136:                     Nvar : Integer;
10137:                     R    : TMatrix); external 'dmath';
10138: { Computes the correlation matrix R from the var-cov matrix V }
10139: procedure PCA(R      : TMatrix;
10140:               Nvar   : Integer;
10141:               Lambda : TVector;
10142:               C, Rc  : TMatrix); external 'dmath';
```

```
10143: { Performs a principal component analysis of the correlation matrix R }
10144: procedure ScaleVar(X            : TMatrix;
10145:                    Lb, Ub, Nvar : Integer;
10146:                    M, S         : TVector;
10147:                    Z            : TMatrix); external 'dmath';
10148: { Scales a set of variables by subtracting means and dividing by SD's }
10149: procedure PrinFac(Z            : TMatrix;
10150:                   Lb, Ub, Nvar : Integer;
10151:                   C, F         : TMatrix); external 'dmath';
10152: { Computes principal factors }
10153: { ---------------------------------------------------------------
10154:   Strings
10155:   --------------------------------------------------------------- }
10156: function LTrim(S : String) : String; external 'dmath';
10157: { Removes leading blanks }
10158: function RTrim(S : String) : String; external 'dmath';
10159: { Removes trailing blanks }
10160: function Trim(S : String) : String; external 'dmath';
10161: { Removes leading and trailing blanks }
10162: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10163: { Returns a string made of character C repeated N times }
10164: function RFill(S : String; L : Byte) : String; external 'dmath';
10165: { Completes string S with trailing blanks for a total length L }
10166: function LFill(S : String; L : Byte) : String; external 'dmath';
10167: { Completes string S with leading blanks for a total length L }
10168: function CFill(S : String; L : Byte) : String; external 'dmath';
10169: { Centers string S on a total length L }
10170: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10171: { Replaces in string S all the occurences of C1 by C2 }
10172: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10173: { Extracts a field from a string }
10174: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10175: { Parses a string into its constitutive fields }
10176: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10177: { Sets the numeric format }
10178: function FloatStr(X : Float) : String; external 'dmath';
10179: { Converts a real to a string according to the numeric format }
10180: function IntStr(N : LongInt) : String; external 'dmath';
10181: { Converts an integer to a string }
10182: function CompStr(Z : Complex) : String; external 'dmath';
10183: { Converts a complex number to a string }
10184: {$IFDEF DELPHI}
10185: function StrDec(S : String) : String; external 'dmath';
10186: { Set decimal separator to the symbol defined in SysUtils }
10187: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10188: { Test if a string represents a number and returns it in X }
10189: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10190: { Reads a floating point number from an Edit control }
10191: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10192: { Writes a floating point number in a text file }
10193: {$ENDIF}
10194: { ---------------------------------------------------------------
10195:   BGI / Delphi graphics
10196:   --------------------------------------------------------------- }
10197: function InitGraphics
10198: {$IFDEF DELPHI}
10199: (Width, Height : Integer) : Boolean;
10200: {$ELSE}
10201: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10202: { Enters graphic mode }
10203: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10204:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10205: { Sets the graphic window }
10206: procedure SetOxScale(Scale               : TScale;
10207:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10208: { Sets the scale on the Ox axis }
10209: procedure SetOyScale(Scale               : TScale;
10210:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10211: { Sets the scale on the Oy axis }
10212: procedure GetOxScale(var Scale                : TScale;
10213:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10214: { Returns the scale on the Ox axis }
10215: procedure GetOyScale(var Scale                : TScale;
10216:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10217: { Returns the scale on the Oy axis }
10218: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10219: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10220: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10221: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10222: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10223: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10224: {$IFNDEF DELPHI}
10225: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10226: { Sets the font for the main graph title }
10227: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10228: { Sets the font for the Ox axis (title and labels) }
10229: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10230: { Sets the font for the Oy axis (title and labels) }
10231: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
```

```
10232: { Sets the font for the legends }
10233: procedure SetClipping(Clip : Boolean); external 'dmath';
10234: { Determines whether drawings are clipped at the current viewport
10235:   boundaries, according to the value of the Boolean parameter Clip }
10236: {$ENDIF}
10237: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10238: { Plots the horizontal axis }
10239: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10240: { Plots the vertical axis }
10241: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10242: { Plots a grid on the graph }
10243: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10244: { Writes the title of the graph }
10245: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10246: { Sets the maximum number of curves and re-initializes their parameters }
10247: procedure SetPointParam
10248: {$IFDEF DELPHI}
10249: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10250: {$ELSE}
10251: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10252: { Sets the point parameters for curve # CurvIndex }
10253: procedure SetLineParam
10254: {$IFDEF DELPHI}
10255: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10256: {$ELSE}
10257: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10258: { Sets the line parameters for curve # CurvIndex }
10259: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10260: { Sets the legend for curve # CurvIndex }
10261: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10262: { Sets the step for curve # CurvIndex }
10263: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10264: procedure GetPointParam
10265: {$IFDEF DELPHI}
10266: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10267: {$ELSE}
10268: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10269: { Returns the point parameters for curve # CurvIndex }
10270: procedure GetLineParam
10271: {$IFDEF DELPHI}
10272: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10273: {$ELSE}
10274: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10275: { Returns the line parameters for curve # CurvIndex }
10276: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10277: { Returns the legend for curve # CurvIndex }
10278: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10279: { Returns the step for curve # CurvIndex }
10280: {$IFDEF DELPHI}
10281: procedure PlotPoint(Canvas    : TCanvas;
10282:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
10283: {$ELSE}
10284: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10285: {$ENDIF}
10286: { Plots a point on the screen }
10287: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10288:                     X, Y                 : TVector;
10289:                     Lb, Ub, CurvIndex    : Integer); external 'dmath';
10290: { Plots a curve }
10291: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10292:                                  X, Y, S              : TVector;
10293:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10294: { Plots a curve with error bars }
10295: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10296:                    Func                 : TFunc;
10297:                    Xmin, Xmax           : Float;
10298:                    {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10299:                    CurvIndex            : Integer); external 'dmath';
10300: { Plots a function }
10301: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10302:                       NCurv              : Integer;
10303:                       ShowPoints, ShowLines : Boolean); external 'dmath';
10304: { Writes the legends for the plotted curves }
10305: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10306:                  Nx, Ny, Nc           : Integer;
10307:                  X, Y, Z              : TVector;
10308:                  F                    : TMatrix); external 'dmath';
10309: { Contour plot }
10310: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10311: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10312: function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
10313: function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
10314: {$IFNDEF DELPHI}
10315: procedure LeaveGraphics; external 'dmath';
10316: { Quits graphic mode }
10317: {$ENDIF}
10318: { ---------------------------------------------------------------
10319:   LaTeX graphics
10320:   --------------------------------------------------------------- }
```

```
10321: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
10322:                           Header            : Boolean) : Boolean; external 'dmath';
10323: { Initializes the LaTeX file }
10324: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10325: { Sets the graphic window }
10326: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10327: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10328: { Sets the scale on the Ox axis }
10329: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10330: { Sets the scale on the Oy axis }
10331: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10332: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10333: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10334: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10335: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10336: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10337: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10338: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10339: { Sets the maximum number of curves and re-initializes their parameters }
10340: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10341: { Sets the point parameters for curve # CurvIndex }
10342: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10343:                            Width : Float; Smooth : Boolean); external 'dmath';
10344: { Sets the line parameters for curve # CurvIndex }
10345: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10346: { Sets the legend for curve # CurvIndex }
10347: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10348: { Sets the step for curve # CurvIndex }
10349: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10350: { Plots a curve }
10351: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10352:                                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10353: { Plots a curve with error bars }
10354: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10355:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
10356: { Plots a function }
10357: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10358: { Writes the legends for the plotted curves }
10359: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
10360: { Contour plot }
10361: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10362: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10363:
10364: //******************************************************unit uPSI_SynPdf;
10365:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10366:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10367:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10368:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10369:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10370:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10371:  //Function _GetCharCount( Text : PAnsiChar) : integer
10372:  //Procedure L2R( W : PWideChar; L : integer)
10373:  Function PdfCoord( MM : single) : integer
10374:  Function CurrentPrinterPaperSize : TPDFPaperSize
10375:  Function CurrentPrinterRes : TPoint
10376:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10377:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10378:  Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10379:  Const('Usp10','String 'usp10.dll
10380:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10381:    'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10382:   AddTypeS('TScriptState_set', 'set of TScriptState_enum
10383: //*********************************************************************
10384:
10385: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10386: begin
10387:  Procedure PMrandomize( I : word)
10388:  Function PMrandom : longint
10389:  Function Rrand : extended
10390:  Function Irand( N : word) : word
10391:  Function Brand( P : extended) : boolean
10392:  Function Nrand : extended
10393: end;
10394:
10395: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10396: begin
10397:   Function Endian( x : LongWord) : LongWord
10398:   Function Endian64( x : Int64) : Int64
10399:   Function spRol( x : LongWord; y : Byte) : LongWord
10400:   Function spRor( x : LongWord; y : Byte) : LongWord
10401:   Function Ror64( x : Int64; y : Byte) : Int64
10402: end;
10403:
10404: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10405: begin
10406:  Procedure ClearModules
10407:  Procedure ReadMapFile( Fname : string)
10408:  Function AddressInfo( Address : dword) : string
10409: end;
```

```
10410:
10411: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10412: begin
10413:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10414:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10415:    +'teByOther, tpExecuteByOther )
10416:   TTarPermissions', 'set of TTarPermission
10417:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10418:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10419:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10420:   TTarModes', 'set of TTarMode
10421:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10422:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10423:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10424:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10425:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10426:   SIRegister_TTarArchive(CL);
10427:   SIRegister_TTarWriter(CL);
10428:  Function PermissionString( Permissions : TTarPermissions) : STRING
10429:  Function ConvertFilename( Filename : STRING) : STRING
10430:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10431:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10432:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10433: end;
10434:
10435:
10436: //************************************************unit uPSI_TlHelp32;
10437: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10438: begin
10439:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10440:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10441:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10442:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10443:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10444:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10445:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10446:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10447:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10448:   AddTypeS('THeapList32', 'tagHEAPLIST32
10449:  Const('HF32_DEFAULT','LongInt'( 1);
10450:  Const('HF32_SHARED','LongInt'( 2);
10451:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10452:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10453:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10454:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10455:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10456:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10457:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10458:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10459:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10460:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10461:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10462:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10463:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10464:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10465:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10466:    +'aPri : Longint; dwFlags : DWORD; end
10467:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10468:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10469:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10470:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10471: end;
10472:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10473:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10474:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10475:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10476:  Const('EWX_LOGOFF','LongInt'( 0);
10477:  Const('EWX_SHUTDOWN','LongInt'( 1);
10478:  Const('EWX_REBOOT','LongInt'( 2);
10479:  Const('EWX_FORCE','LongInt'( 4);
10480:  Const('EWX_POWEROFF','LongInt'( 8);
10481:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10482:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10483:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10484:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10485:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10486:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10487:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10488:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10489:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10490:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10491:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10492:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10493:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10494:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10495:  Function GetDesktopWindow : HWND
10496:  Function GetParent( hWnd : HWND) : HWND
10497:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10498:  Function GetTopWindow( hWnd : HWND) : HWND
```

```
10499:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10500:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10501:  //Delphi DFM
10502:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10503:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10504:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10505:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10506:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10507:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10508:  Function OpenIcon( hWnd : HWND) : BOOL
10509:  Function CloseWindow( hWnd : HWND) : BOOL
10510:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10511:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10512:  Function IsWindowVisible( hWnd : HWND) : BOOL
10513:  Function IsIconic( hWnd : HWND) : BOOL
10514:  Function AnyPopup : BOOL
10515:  Function BringWindowToTop( hWnd : HWND) : BOOL
10516:  Function IsZoomed( hWnd : HWND) : BOOL
10517:  Function IsWindow( hWnd : HWND) : BOOL
10518:  Function IsMenu( hMenu : HMENU) : BOOL
10519:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10520:  Function DestroyWindow( hWnd : HWND) : BOOL
10521:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10522:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10523:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10524:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10525:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10526:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10527:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10528:
10529: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10530: begin
10531:  const('ShowSetupDialogOptLong','String '--setup
10532:  PrimaryConfPathOptLong','String '--primary-config-path=
10533:  PrimaryConfPathOptShort','String '--pcp=
10534:  SecondaryConfPathOptLong','String '--secondary-config-path=
10535:  SecondaryConfPathOptShort','String '--scp=
10536:  NoSplashScreenOptLong','String '--no-splash-screen
10537:  NoSplashScreenOptShort','String '--nsc
10538:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10539:  SkipLastProjectOpt','String '--skip-last-project
10540:  DebugLogOpt','String '--debug-log=
10541:  DebugLogOptEnable','String '--debug-enable=
10542:  LanguageOpt','String '--language=
10543:  LazarusDirOpt','String '--lazarusdir=
10544:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10545:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10546:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10547:  Function IsHelpRequested : Boolean
10548:  Function IsVersionRequested : boolean
10549:  Function GetLanguageSpecified : string
10550:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10551:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10552:  Procedure ParseNoGuiCmdLineParams
10553:  Function ExtractCmdLineFilenames : TStrings
10554: end;
10555:
10556:
10557: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10558: begin
10559:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10560:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10561:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10562:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10563:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10564:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10565:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10566:  Function DirPathExists( DirectoryName : string) : boolean
10567:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10568:  Function ExtractFileNameOnly( const AFilename : string) : string
10569:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10570:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10571:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10572:  Function ForceDirectory( DirectoryName : string) : boolean
10573:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10574:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10575:  Function FileIsText( const AFilename : string) : boolean
10576:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10577:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10578:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10579:  Function TrimFilename( const AFilename : string) : string
10580:  Function ResolveDots( const AFilename : string) : string
10581:  Procedure ForcePathDelims( var FileName : string)
10582:  Function GetForcedPathDelims( const FileName : string) : String
10583:  Function CleanAndExpandFilename( const Filename : string) : string
10584:  Function CleanAndExpandDirectory( const Filename : string) : string
10585:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10586:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10587:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
```

```
10588:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
10589:  Function FileIsInPath( const Filename, Path : string) : boolean
10590:  Function AppendPathDelim( const Path : string) : string
10591:  Function ChompPathDelim( const Path : string) : string
10592:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10593:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10594:  Function MinimizeSearchPath( const SearchPath : string) : string
10595:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10596:  (*Function FileExistsUTF8( const Filename : string) : boolean
10597:  Function FileAgeUTF8( const FileName : string) : Longint
10598:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10599:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10600:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10601:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10602:  Procedure FindCloseUTF8( var F : TSearchrec)
10603:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10604:  Function FileGetAttrUTF8( const FileName : String) : Longint
10605:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10606:  Function DeleteFileUTF8( const FileName : String) : Boolean
10607:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10608:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10609:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10610:  Function GetCurrentDirUTF8 : String
10611:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10612:  Function CreateDirUTF8( const NewDir : String) : Boolean
10613:  Function RemoveDirUTF8( const Dir : String) : Boolean
10614:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10615:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10616:  Function FileCreateUTF8( const FileName : string) : THandle;
10617:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10618:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10619:  Function FileSizeUtf8( const Filename : string) : int64
10620:  Function GetFileDescription( const AFilename : string) : string
10621:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10622:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10623:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10624:  Function IsUNCPath( const Path : String) : Boolean
10625:  Function ExtractUNCVolume( const Path : String) : String
10626:  Function ExtractFileRoot( FileName : String) : String
10627:  Function GetDarwinSystemFilename( Filename : string) : string
10628:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10629:  Function StrToCmdLineParam( const Param : string) : string
10630:  Function MergeCmdLineParams( ParamList : TStrings) : string
10631:  Procedure InvalidateFileStateCache( const Filename : string)
10632:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10633:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10634:  Function FindAllDocs(const Root, extmask: string): TStringlist;
10635:  Function ReadFileToString( const Filename : string) : string
10636:  procedure Inc1(var X: longint; N: Longint);
10637:
10638:  type
10639:   TCopyFileFlag = ( cffOverwriteFile,
10640:     cffCreateDestDirectory, cffPreserveTime );
10641:   TCopyFileFlags = set of TCopyFileFlag;*)
10642:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10643:   TCopyFileFlags', 'set of TCopyFileFlag
10644:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10645: end;
10646:
10647: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10648: begin
10649:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10650:   SIRegister_TMask(CL);
10651:   SIRegister_TParseStringList(CL);
10652:   SIRegister_TMaskList(CL);
10653:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10654:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10655:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10656:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10657: end;
10658:
10659: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10660: begin
10661:   //PShellHookInfo', '^TShellHookInfo // will not work
10662:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10663:   SHELLHOOKINFO', 'TShellHookInfo
10664:   LPSHELLHOOKINFO', 'PShellHookInfo
10665:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10666:   SIRegister_TJvShellHook(CL);
10667:  Function InitJvShellHooks : Boolean
10668:  Procedure UnInitJvShellHooks
10669: end;
10670:
10671: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10672: begin
10673:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10674:    +', dcHasSetSel, dcWantTab, dcNative )
10675:   TDlgCodes', 'set of TDlgCode
```

```
10676:  'dcWantMessage',' dcWantAllKeys);
10677:   SIRegister_IJvExControl(CL);
10678:   SIRegister_IJvDenySubClassing(CL);
10679:   SIRegister_TStructPtrMessage(CL);
10680:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10681:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10682:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10683:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10684:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10685:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10686:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10687:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10688:  Function GetFocusedControl( AControl : TControl) : TWinControl
10689:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10690:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10691:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10692:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10693:   SIRegister_TJvExControl(CL);
10694:   SIRegister_TJvExWinControl(CL);
10695:   SIRegister_TJvExCustomControl(CL);
10696:   SIRegister_TJvExGraphicControl(CL);
10697:   SIRegister_TJvExHintWindow(CL);
10698:   SIRegister_TJvExPubGraphicControl(CL);
10699: end;
10700:
10701: (*----------------------------------------------------------------------------*)
10702: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10703: begin
10704:  Procedure EncodeStream( Input, Output : TStream)
10705:  Procedure DecodeStream( Input, Output : TStream)
10706:  Function EncodeString1( const Input : string) : string
10707:  Function DecodeString1( const Input : string) : string
10708: end;
10709:
10710: (*----------------------------------------------------------------------------*)
10711: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10712: begin
10713:   SIRegister_TWebAppRegInfo(CL);
10714:   SIRegister_TWebAppRegList(CL);
10715:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10716:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10717:  Procedure UnregisterWebApp( const AProgID : string)
10718:  Function FindRegisteredWebApp( const AProgID : string) : string
10719:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10720:  'sUDPPort','String 'UDPPort
10721: end;
10722:
10723: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10724: begin
10725:  // TStringDynArray', 'array of string
10726:  Function GetEnvVarValue( const VarName : string) : string
10727:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10728:  Function DeleteEnvVar( const VarName : string) : Integer
10729:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10730:  Function ExpandEnvVars( const Str : string) : string
10731:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10732:  Procedure GetAllEnvVarNames( const Names : TStrings);
10733:  Function GetAllEnvVarNames1 : TStringDynArray;
10734:  Function EnvBlockSize : Integer
10735:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10736:   SIRegister_TPJEnvVarsEnumerator(CL);
10737:   SIRegister_TPJEnvVars(CL);
10738:   FindClass('TOBJECT'),'EPJEnvVars
10739:   FindClass('TOBJECT'),'EPJEnvVars
10740:  //Procedure Register
10741: end;
10742:
10743: (*----------------------------------------------------------------------------*)
10744: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10745: begin
10746:  'cOneSecInMS','LongInt'( 1000);
10747:  //'cDefTimeSlice','LongInt'( 50);
10748:  //'cDefMaxExecTime',' cOneMinInMS);
10749:  'cAppErrorMask','LongInt'( 1 shl 29);
10750:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10751:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10752:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10753:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10754:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10755:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10756:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10757:   SIRegister_TPJCustomConsoleApp(CL);
10758:   SIRegister_TPJConsoleApp(CL);
10759: end;
10760:
10761: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10762: begin
10763:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
```

```
10764:   t_encoding', '( uuencode, base64, mime )
10765:  Function internet_date( date : TDateTime) : string
10766:  Function lookup_hostname( const hostname : string) : longint
10767:  Function my_hostname : string
10768:  Function my_ip_address : longint
10769:  Function ip2string( ip_address : longint) : string
10770:  Function resolve_hostname( ip : longint) : string
10771:  Function address_from( const s : string; count : integer) : string
10772:  Function encode_base64( data : TStream) : TStringList
10773:  Function decode_base64( source : TStringList) : TMemoryStream
10774:  Function posn( const s, t : string; count : integer) : integer
10775:  Function poscn( c : char; const s : string; n : integer) : integer
10776:  Function filename_of( const s : string) : string
10777:  //Function trim( const s : string) : string
10778:  //Procedure setlength( var s : string; l : byte)
10779:  Function TimeZoneBias : longint
10780:  Function eight2seven_quoteprint( const s : string) : string
10781:  Function eight2seven_german( const s : string) : string
10782:  Function seven2eight_quoteprint( const s : string) : string end;
10783:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10784:  Function socketerror : cint
10785:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10786:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10787:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10788:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10789:  Function fplisten( s : cint; backlog : cint) : cint
10790:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10791:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10792:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10793:  Function NetAddrToStr( Entry : in_addr) : String
10794:  Function HostAddrToStr( Entry : in_addr) : String
10795:  Function StrToHostAddr( IP : String) : in_addr
10796:  Function StrToNetAddr( IP : String) : in_addr
10797:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10798:   cint8', 'shortint
10799:   cuint8', 'byte
10800:   cchar', 'cint8
10801:   cschar', 'cint8
10802:   cuchar', 'cuint8
10803:   cint16', 'smallint
10804:   cuint16', 'word
10805:   cshort', 'cint16
10806:   csshort', 'cint16
10807:   cushort', 'cuint16
10808:   cint32', 'longint
10809:   cuint32', 'longword
10810:   cint', 'cint32
10811:   csint', 'cint32
10812:   cuint', 'cuint32
10813:   csigned', 'cint
10814:   cunsigned', 'cuint
10815:   cint64', 'int64
10816:   clonglong', 'cint64
10817:   cslonglong', 'cint64
10818:   cbool', 'longbool
10819:   cfloat', 'single
10820:   cdouble', 'double
10821:   clongdouble', 'extended
10822:
10823: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10824: begin
10825:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10826:   SIRegister_TlkJSONdotnetclass(CL);
10827:   SIRegister_TlkJSONbase(CL);
10828:   SIRegister_TlkJSONnumber(CL);
10829:   SIRegister_TlkJSONstring(CL);
10830:   SIRegister_TlkJSONboolean(CL);
10831:   SIRegister_TlkJSONnull(CL);
10832:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10833:    +'se; data : TObject; var Continue : Boolean)
10834:   SIRegister_TlkJSONcustomlist(CL);
10835:   SIRegister_TlkJSONlist(CL);
10836:   SIRegister_TlkJSONobjectmethod(CL);
10837:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10838:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10839:   SIRegister_TlkHashTable(CL);
10840:   SIRegister_TlkBalTree(CL);
10841:   SIRegister_TlkJSONobject(CL);
10842:   SIRegister_TlkJSON(CL);
10843:   SIRegister_TlkJSONstreamed(CL);
10844:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10845: end;
10846:
10847: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10848: begin
10849:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10850:   SIRegister_TZSortedList(CL);
10851:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10852:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
```

```
10853:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10854:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10855:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10856:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10857:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10858:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10859:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10860:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10861:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10862:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10863:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10864:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10865:  Function StrToBoolEx( Str : string) : Boolean
10866:  Function BoolToStrEx( Bool : Boolean) : String
10867:  Function IsIpAddr( const Str : string) : Boolean
10868:  Function zSplitString( const Str, Delimiters : string) : TStrings
10869:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10870:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10871:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10872:  Function FloatToSQLStr( Value : Extended) : string
10873:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10874:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10875:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10876:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10877:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10878:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10879:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10880:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10881:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10882:  Function BytesToVar( const Value : TByteDynArray) : Variant
10883:  Function VarToBytes( const Value : Variant) : TByteDynArray
10884:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10885:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10886:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10887:  Function EncodeCString( const Value : string) : string
10888:  Function DecodeCString( const Value : string) : string
10889:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10890:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10891:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
10892:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
       SubVersion:Integer):Int;
10893:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10894:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10895:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10896:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10897:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10898:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10899:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10900:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10901:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10902:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10903: end;
10904:
10905: unit uPSI_ZEncoding;
10906:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10907:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10908:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10909:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10910:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10911:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10912:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10913:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10914:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10915:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10916:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10917:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10918:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10919:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10920:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10921:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10922:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10923:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10924:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10925:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10926:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10927:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10928:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10929:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10930:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10931:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10932:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10933:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10934:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10935:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10936:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10937:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10938:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10939:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
```

```
10940:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10941:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10942:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10943:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10944:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10945:  Function ZDefaultSystemCodePage : Word
10946:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10947:
10948:
10949: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10950: begin
10951:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10952:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10953:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10954:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10955:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10956:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10957:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10958:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10959:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10960:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10961:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
10962:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10963:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10964:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10965:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10966:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10967:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10968:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10969:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10970:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10971:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10972:  ('ilIdentity','2 RPC_C_IMP_LEVEL_IDENTIFY);
10973:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10974:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10975:  ('EOAC_NONE','LongWord').SetUInt( $0);
10976:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10977:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10978:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10979:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10980:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10981:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10982:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10983:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10984:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10985:   FindClass('TOBJECT'),'EBoldCom
10986:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10987:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10988:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10989:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10990:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10991:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10992:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10993:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10994:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10995:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10996:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10997:  Function BoldCreateGUID : TGUID
10998:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
10999:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
11000:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
11001:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
11002: end;
11003:
11004: (*----------------------------------------------------------------------------*)
11005: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
11006: begin
11007:  Function ParseISODate( s : string) : TDateTime
11008:  Function ParseISODateTime( s : string) : TDateTime
11009:  Function ParseISOTime( str : string) : TDateTime
11010: end;
11011:
11012: (*----------------------------------------------------------------------------*)
11013: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11014: begin
11015:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
11016:  Function BoldCreateGUIDWithBracketsAsString : string
11017: end;
11018:
11019: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11020: begin
11021:   FindClass('TOBJECT'),'TBoldFileHandler
11022:   FindClass('TOBJECT'),'TBoldDiskFileHandler
11023:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11024:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
11025:   SIRegister_TBoldFileHandler(CL);
11026:   SIRegister_TBoldDiskFileHandler(CL);
11027:  Procedure BoldCloseAllFilehandlers
```

```
11028:  Procedure BoldRemoveUnchangedFilesFromEditor
11029:  Function BoldFileHandlerList : TBoldObjectArray
11030:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
        OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11031: end;
11032:
11033: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11034: begin
11035:   PCharArr', 'array of PChar
11036:  Function BoldInternetOpen(Agent:String;
        AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
11037:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11038:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
        NumberOfBytesRead:Card):LongBool;
11039:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
11040:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
        Cardinal; Reserved : Cardinal) : LongBool
11041:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
        Cardinal; Context : Cardinal) : LongBool
11042:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
        : PCharArr; Flags, Context : Cardinal) : Pointer
11043:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11044:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11045:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11046:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
        Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11047:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11048: end;
11049:
11050: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11051: begin
11052:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11053:   SIRegister_TfrmBoldQueryUser(CL);
11054:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
11055: end;
11056:
11057: (*----------------------------------------------------------------------------*)
11058: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11059: begin
11060:  //('befIsInDisplayList',' BoldElementFlag0);
11061:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11062:  //('befFollowerSelected',' BoldElementFlag2);
11063:   FindClass('TOBJECT'),'TBoldQueue
11064:   FindClass('TOBJECT'),'TBoldQueueable
11065:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11066:   SIRegister_TBoldQueueable(CL);
11067:   SIRegister_TBoldQueue(CL);
11068:  Function BoldQueueFinalized : Boolean
11069:  Function BoldInstalledQueue : TBoldQueue
11070: end;
11071:
11072: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
11073: begin
11074:  const mmPerInch','Extended').setExtended( 25.4);
11075:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11076:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11077:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11078:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11079:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11080:   TBarLineType', '( white, black, black_half )
11081:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11082:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11083:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11084:   TCheckSumMethod', '( csmNone, csmModulo10 )
11085:   SIRegister_TAsBarcode(CL);
11086:  Function CheckSumModulo10( const data : string) : string
11087:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11088:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11089:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11090:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11091: end;
11092:
11093: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11094: begin
11095:    THomogeneousByteVector', 'array[0..3] of Byte
11096:   THomogeneousWordVector', 'array[0..3] of Word
11097:   THomogeneousIntVector', 'array[0..3] of Integer
11098:   THomogeneousFltVector', 'array[0..3] of single
11099:   THomogeneousDblVector', 'array[0..3] of double
11100:   THomogeneousExtVector', 'array[0..3] of extended
11101:   TAffineByteVector', 'array[0..2] of Byte
11102:   TAffineWordVector', 'array[0..2] of Word
11103:   TAffineIntVector', 'array[0..2] of Integer
11104:   TAffineFltVector', 'array[0..2] of single
11105:   TAffineDblVector', 'array[0..2] of double
11106:   TAffineExtVector', 'array[0..2] of extended
11107:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11108:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11109:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
```

```
11110:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11111:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11112:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11113:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11114:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11115:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11116:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11117:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11118:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11119:   TMatrix4b', 'THomogeneousByteMatrix
11120:   TMatrix4w', 'THomogeneousWordMatrix
11121:   TMatrix4i', 'THomogeneousIntMatrix
11122:   TMatrix4f', 'THomogeneousFltMatrix
11123:   TMatrix4d', 'THomogeneousDblMatrix
11124:   TMatrix4e', 'THomogeneousExtMatrix
11125:   TMatrix3b', 'TAffineByteMatrix
11126:   TMatrix3w', 'TAffineWordMatrix
11127:   TMatrix3i', 'TAffineIntMatrix
11128:   TMatrix3f', 'TAffineFltMatrix
11129:   TMatrix3d', 'TAffineDblMatrix
11130:   TMatrix3e', 'TAffineExtMatrix
11131:   //'PMatrix', '^TMatrix // will not work
11132:   TMatrixGL', 'THomogeneousFltMatrix
11133:   THomogeneousMatrix', 'THomogeneousFltMatrix
11134:   TAffineMatrix', 'TAffineFltMatrix
11135:   TQuaternion', 'record Vector : TVector4f; end
11136:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11137:   +'ger; Height : Integer; end
11138:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11139:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11140:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11141:  'EPSILON','Extended').setExtended( 1E-100);
11142:  'EPSILON2','Extended').setExtended( 1E-50);
11143:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11144:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11145:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11146:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11147:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11148:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11149:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11150:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11151:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11152:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11153:  Function VectorLength( V : array of Single) : Single
11154:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11155:  Procedure VectorNegate( V : array of Single)
11156:  Function VectorNorm( V : array of Single) : Single
11157:  Function VectorNormalize( V : array of Single) : Single
11158:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11159:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11160:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11161:  Procedure VectorScale( V : array of Single; Factor : Single)
11162:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11163:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11164:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11165:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11166:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11167:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11168:  Procedure MatrixAdjoint( var M : TMatrixGL)
11169:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11170:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11171:  Function MatrixDeterminant( M : TMatrixGL) : Single
11172:  Procedure MatrixInvert( var M : TMatrixGL)
11173:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11174:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11175:  Procedure MatrixTranspose( var M : TMatrixGL)
11176:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11177:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11178:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11179:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11180:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11181:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11182:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11183:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11184:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11185:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11186:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11187:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11188:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11189:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11190:  Function MakeAffineVector( V : array of Single) : TAffineVector
11191:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11192:  Function MakeVector( V : array of Single) : TVectorGL
11193:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11194:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11195:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11196:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11197:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11198:  Function ArcCosGL( X : Extended) : Extended
```

```
11199:  Function ArcSinGL( X : Extended) : Extended
11200:  Function ArcTan2GL( Y, X : Extended) : Extended
11201:  Function CoTanGL( X : Extended) : Extended
11202:  Function DegToRadGL( Degrees : Extended) : Extended
11203:  Function RadToDegGL( Radians : Extended) : Extended
11204:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11205:  Function TanGL( X : Extended) : Extended
11206:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11207:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11208:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11209:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11210:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11211:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11212: end;
11213:
11214:
11215: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11216: begin
11217:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11218:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11219:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11220:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11221:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11222:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11223:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11224:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11225:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11226:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11227:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11228:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11229:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11230:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11231:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11232:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11233:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11234:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11235:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11236:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11237:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11238:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11239:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11240:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11241:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
      Items:TStrings):Bool;
11242:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
      SaveTo:TStrings):Bool;
11243:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11244: end;
11245:
11246: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11247: begin
11248:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11249:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11250:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11251:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11252:   FindClass('TOBJECT'),'EInvalidParam
11253:  Function IsDCOMInstalled : Boolean
11254:  Function IsDCOMEnabled : Boolean
11255:  Function GetDCOMVersion : string
11256:  Function GetMDACVersion : string
11257:  Function GetMDACVersion2 : string
11258:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
      VarArray:OleVariant):HResult;
11259:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11260:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
      VarArray:OleVariant):HResult;
11261:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11262:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
      VarArray:OleVariant):HResult;
11263:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11264:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11265:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11266:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11267:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11268:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11269:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11270:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11271:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11272: end;
11273:
11274:
11275: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11276: begin
11277:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11278:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11279:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11280:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11281:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11282:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
```

```
11283:  DegPerCycle','Extended').setExtended( 360.0);
11284:  DegPerGrad','Extended').setExtended( 0.9);
11285:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11286:  GradPerCycle','Extended').setExtended( 400.0);
11287:  GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
11288:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11289:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11290:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11291:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11292:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11293:  CyclePerGrad','Extended').setExtended( 0.0025);
11294:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11295:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11296:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11297:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11298:  Function MakePercentage( const Step, Max : Longint) : Longint
11299:  Function CelsiusToKelvin( const T : double) : double
11300:  Function CelsiusToFahrenheit( const T : double) : double
11301:  Function KelvinToCelsius( const T : double) : double
11302:  Function KelvinToFahrenheit( const T : double) : double
11303:  Function FahrenheitToCelsius( const T : double) : double
11304:  Function FahrenheitToKelvin( const T : double) : double
11305:  Function CycleToDeg( const Cycles : double) : double
11306:  Function CycleToGrad( const Cycles : double) : double
11307:  Function CycleToRad( const Cycles : double) : double
11308:  Function DegToCycle( const Degrees : double) : double
11309:  Function DegToGrad( const Degrees : double) : double
11310:  Function DegToRad( const Degrees : double) : double
11311:  Function GradToCycle( const Grads : double) : double
11312:  Function GradToDeg( const Grads : double) : double
11313:  Function GradToRad( const Grads : double) : double
11314:  Function RadToCycle( const Radians : double) : double
11315:  Function RadToDeg( const Radians : double) : double
11316:  Function RadToGrad( const Radians : double) : double
11317:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11318:  Function DmsToRad( const D, M : Integer; const S : double) : double
11319:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11320:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11321:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11322:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11323:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11324:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11325:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11326:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11327:  Function CmToInch( const Cm : double) : double
11328:  Function InchToCm( const Inch : double) : double
11329:  Function FeetToMetre( const Feet : double) : double
11330:  Function MetreToFeet( const Metre : double) : double
11331:  Function YardToMetre( const Yard : double) : double
11332:  Function MetreToYard( const Metre : double) : double
11333:  Function NmToKm( const Nm : double) : double
11334:  Function KmToNm( const Km : double) : double
11335:  Function KmToSm( const Km : double) : double
11336:  Function SmToKm( const Sm : double) : double
11337:  Function LitreToGalUs( const Litre : double) : double
11338:  Function GalUsToLitre( const GalUs : double) : double
11339:  Function GalUsToGalCan( const GalUs : double) : double
11340:  Function GalCanToGalUs( const GalCan : double) : double
11341:  Function GalUsToGalUk( const GalUs : double) : double
11342:  Function GalUkToGalUs( const GalUk : double) : double
11343:  Function LitreToGalCan( const Litre : double) : double
11344:  Function GalCanToLitre( const GalCan : double) : double
11345:  Function LitreToGalUk( const Litre : double) : double
11346:  Function GalUkToLitre( const GalUk : double) : double
11347:  Function KgToLb( const Kg : double) : double
11348:  Function LbToKg( const Lb : double) : double
11349:  Function KgToOz( const Kg : double) : double
11350:  Function OzToKg( const Oz : double) : double
11351:  Function CwtUsToKg( const Cwt : double) : double
11352:  Function CwtUkToKg( const Cwt : double) : double
11353:  Function KaratToKg( const Karat : double) : double
11354:  Function KgToCwtUs( const Kg : double) : double
11355:  Function KgToCwtUk( const Kg : double) : double
11356:  Function KgToKarat( const Kg : double) : double
11357:  Function KgToSton( const Kg : double) : double
11358:  Function KgToLton( const Kg : double) : double
11359:  Function StonToKg( const STon : double) : double
11360:  Function LtonToKg( const Lton : double) : double
11361:  Function QrUsToKg( const Qr : double) : double
11362:  Function QrUkToKg( const Qr : double) : double
11363:  Function KgToQrUs( const Kg : double) : double
11364:  Function KgToQrUk( const Kg : double) : double
11365:  Function PascalToBar( const Pa : double) : double
11366:  Function PascalToAt( const Pa : double) : double
11367:  Function PascalToTorr( const Pa : double) : double
11368:  Function BarToPascal( const Bar : double) : double
11369:  Function AtToPascal( const At : double) : double
11370:  Function TorrToPascal( const Torr : double) : double
11371:  Function KnotToMs( const Knot : double) : double
```

```
11372:  Function HpElectricToWatt( const HpE : double) : double
11373:  Function HpMetricToWatt( const HpM : double) : double
11374:  Function MsToKnot( const ms : double) : double
11375:  Function WattToHpElectric( const W : double) : double
11376:  Function WattToHpMetric( const W : double) : double
11377:  function getBigPI: string;    //PI of 1000 numbers
11378:
11379: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11380: begin
11381:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11382:  Procedure CDCopyFile( const FileName, DestName : string)
11383:  Procedure CDMoveFile( const FileName, DestName : string)
11384:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11385:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11386:  Function CDGetTempDir : string
11387:  Function CDGetFileSize( FileName : string) : longint
11388:  Function GetFileTime( FileName : string) : longint
11389:  Function GetShortName( FileName : string) : string
11390:  Function GetFullName( FileName : string) : string
11391:  Function WinReboot : boolean
11392:  Function WinDir : String
11393:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11394:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11395:  Function devExecutor : TdevExecutor
11396: end;
11397:
11398: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11399: begin
11400:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11401:  Procedure Associate( Index : integer)
11402:  Procedure UnAssociate( Index : integer)
11403:  Function IsAssociated( Index : integer) : boolean
11404:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11405:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11406:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11407:  procedure RefreshIcons;
11408: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11409: function MergColor(Colors: Array of TColor): TColor;
11410: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11411: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11412: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11413: function GetInverseColor(AColor: TColor): TColor;
11414: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11415: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11416: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11417: Procedure GetSystemMenuFont(Font: TFont);
11418: end;
11419:
11420: //*************************unit uPSI_JvHLParser;*****************************
11421: function IsStringConstant(const St: string): Boolean;
11422: function IsIntConstant(const St: string): Boolean;
11423: function IsRealConstant(const St: string): Boolean;
11424: function IsIdentifier(const ID: string): Boolean;
11425: function GetStringValue(const St: string): string;
11426: procedure ParseString(const S: string; Ss: TStrings);
11427: function IsStringConstantW(const St: WideString): Boolean;
11428: function IsIntConstantW(const St: WideString): Boolean;
11429: function IsRealConstantW(const St: WideString): Boolean;
11430: function IsIdentifierW(const ID: WideString): Boolean;
11431: function GetStringValueW(const St: WideString): WideString;
11432: procedure ParseStringW(const S: WideString; Ss: TStrings);
11433:
11434:
11435: //*************************unit uPSI_JclMapi;*****************************
11436:
11437: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11438: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11439: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11440: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11441: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11442:
11443: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11444: begin
11445:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11446:  Function BuildType1Message( ADomain, AHost : String) : String
11447:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11448:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11449:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11450: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11451: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11452: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11453: end;
11454:
11455: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11456: begin
11457: ('IpAny','LongWord').SetUInt( $00000000);
```

```
11458:  IpLoopBack','LongWord').SetUInt( $7F000001);
11459:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11460:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11461:  PortAny','LongWord( $0000);
11462:  SocketMaxConnections','LongInt'( 5);
11463:  TIpAddr', 'LongWord
11464:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11465:  Function HostToNetLong( HostLong : LongWord) : LongWord
11466:  Function HostToNetShort( HostShort : Word) : Word
11467:  Function NetToHostLong( NetLong : LongWord) : LongWord
11468:  Function NetToHostShort( NetShort : Word) : Word
11469:  Function StrToIp( Ip : string) : TIpAddr
11470:  Function IpToStr( Ip : TIpAddr) : string
11471: end;
11472:
11473: (*----------------------------------------------------------------------------*)
11474: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11475: begin
11476:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11477:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11478:    +'amSha1, AlsmtpClientAuthAutoSelect )
11479:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11480:   SIRegister_TAlSmtpClient(CL);
11481: end;
11482:
11483: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11484: begin
11485: 'TBitNo', 'Integer
11486:   TStByteNo', 'Integer
11487: TStationNo', 'Integer
11488: TInOutNo', 'Integer
11489: TIo', '( EE, AA, NE, NA )
11490: TBitSet', 'set of TBitNo
11491: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11492: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11493: TBitAddr', 'LongInt
11494: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11495: TByteAddr', 'SmallInt
11496: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11497:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11498:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11499:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11500:  Function BitAddrToStr( Value : TBitAddr) : string
11501:  Function StrToBitAddr( const Value : string) : TBitAddr
11502:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11503:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11504:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11505:  Function ByteAddrToStr( Value : TByteAddr) : string
11506:  Function StrToByteAddr( const Value : string) : TByteAddr
11507:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11508:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11509:  Function InOutStateToStr( State : TInOutState) : string
11510:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11511:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11512: end;
11513:
11514: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11515: begin
11516: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11517:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11518: DpmiPmVector', 'Int64
11519:  'DInterval','LongInt'( 1000);
11520:  //'DEnabled','Boolean')BoolToStr( True);
11521:  'DIntFreq','string' if64
11522:  //'DMessages','Boolean if64);
11523:   SIRegister_TwdxCustomTimer(CL);
11524:   SIRegister_TwdxTimer(CL);
11525:   SIRegister_TwdxRtcTimer(CL);
11526:   SIRegister_TCustomIntTimer(CL);
11527:   SIRegister_TIntTimer(CL);
11528:   SIRegister_TRtcIntTimer(CL);
11529:  Function RealNow : TDateTime
11530:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11531:  Function DateTimeToMs( Time : TDateTime) : LongInt
11532: end;
11533:
11534: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11535: begin
11536: TIdSyslogPRI', 'Integer
11537: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11538:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11539:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11540:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11541:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11542: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11543:   SIRegister_TIdSysLogMsgPart(CL);
11544:   SIRegister_TIdSysLogMessage(CL);
11545:  Function FacilityToString( AFac : TIdSyslogFacility) : string
```

```
11546:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11547:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11548:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11549:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11550:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11551: end;
11552:
11553: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11554: begin
11555:  'UWhitespace','String '(?:\s*)
11556:  Function StripSpaces( const AText : string) : string
11557:  Function CharCount( const AText : string; Ch : Char) : Integer
11558:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11559:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11560: end;
11561:
11562:
11563: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11564: begin
11565:  ExtPascalVersion','String '0.9.8
11566:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11567:    +'Opera, brKonqueror, brMobileSafari )
11568:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11569:   AddTypeS('TExtProcedure', 'Procedure
11570:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11571:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11572:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11573:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11574:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11575:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11576:  Function StrToJS( const S : string; UseBR : boolean) : string
11577:  Function CaseOf( const S : string; const Cases : array of string) : integer
11578:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11579:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11580:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11581:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11582:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11583:  Function IsUpperCase( S : string) : boolean
11584:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11585:  Function BeautifyCSS( const AStyle : string) : string
11586:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11587:  Function JSDateToDateTime( JSDate : string) : TDateTime
11588: end;
11589:
11590: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11591: begin
11592:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11593:   TSHDeleteOptions', 'set of TSHDeleteOption
11594:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11595:   TSHRenameOptions', 'set of TSHRenameOption
11596:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11597:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11598:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11599:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11600:   TEnumFolderFlags', 'set of TEnumFolderFlag
11601:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11602:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11603:    +'IEnumIdList; Folder : IShellFolder; end
11604:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11605:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11606:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11607:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11608:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11609:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11610:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11611:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11612:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11613:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11614:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11615:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11616:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11617:  Function SHFreeMem( var P : Pointer) : Boolean
11618:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11619:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11620:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11621:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11622:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11623:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11624:  Function PidlFree( var IdList : PItemIdList) : Boolean
11625:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11626:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11627:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11628:  Function PidlToPath( IdList : PItemIdList) : string
11629:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11630:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11631:   PShellLink', '^TShellLink // will not work
11632:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11633:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11634:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
```

```
11635:  Procedure ShellLinkFree( var Link : TShellLink)
11636:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11637:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11638:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11639:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11640:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11641:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11642:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11643:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11644:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11645:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11646:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11647:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11648:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11649:  Function ShellOpenAs( const FileName : string) : Boolean
11650:  Function ShellRasDial( const EntryName : string) : Boolean
11651:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11652:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11653:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11654:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11655:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11656:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11657:  Function OemKeyScan( wOemChar : Word) : DWORD
11658:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11659: end;
11660:
11661: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11662: begin
11663:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11664:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11665:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11666:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11667:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11668:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11669:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11670:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11671:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11672:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11673:  Function xmlValidName( const Text : UnicodeString) : Boolean
11674:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11675:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11676:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11677:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11678:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11679:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11680:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11681:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11682:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11683:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11684:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11685:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11686:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11687:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11688:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11689:  Procedure SelfTestcXMLFunctions
11690: end;
11691:
11692: (*----------------------------------------------------------------------------*)
11693: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11694: begin
11695:  Function AWaitCursor : IUnknown
11696:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11697:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11698:  Function YesNo( const ACaption, AMsg : string) : boolean
11699:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11700:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11701:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11702:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11703:  Procedure GetSystemPaths( Strings : TStrings)
11704:  Procedure MakeEditNumeric( EditHandle : integer)
11705: end;
11706:
11707: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11708: begin
11709:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11710:  'BI_YUY2','LongWord( $32595559);
11711:  'BI_UYVY','LongWord').SetUInt( $59565955);
11712:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11713:  'BI_YVU9','LongWord').SetUInt( $39555659);
11714:  'BI_YUV12','LongWord( $30323449);
11715:  'BI_Y8','LongWord').SetUInt( $20203859);
11716:  'BI_Y211','LongWord').SetUInt( $31313259);
11717:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11718:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11719: end;
11720:
11721: (*----------------------------------------------------------------------------*)
11722: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
```

```
11723: begin
11724:  'WM_USER','LongWord').SetUInt( $0400);
11725:  'WM_CAP_START','LongWord').SetUint($0400);
11726:  'WM_CAP_END','longword').SetUint($0400+85);
11727:  //WM_CAP_START+  85
11728:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11729:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11730:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11731:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11732:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11733:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11734:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11735:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11736:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11737:  Function capGetUserData( hwnd : THandle) : LongInt
11738:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11739:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11740:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11741:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11742:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11743:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11744:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11745:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11746:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11747:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11748:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11749:  Function capEditCopy( hwnd : THandle) : LongInt
11750:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11751:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11752:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11753:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11754:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11755:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11756:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11757:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11758:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11759:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11760:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11761:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11762:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11763:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11764:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11765:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11766:  Function capGrabFrame( hwnd : THandle) : LongInt
11767:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11768:  Function capCaptureSequence( hwnd : THandle) : LongInt
11769:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11770:  Function capCaptureStop( hwnd : THandle) : LongInt
11771:  Function capCaptureAbort( hwnd : THandle) : LongInt
11772:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11773:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11774:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11775:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11776:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11777:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11778:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11779:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11780:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11781:  Function capPalettePaste( hwnd : THandle) : LongInt
11782:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11783:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11784:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11785:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11786:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11787:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11788:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11789:   //PCapStatus', '^TCapStatus // will not work
11790:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11791:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11792:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11793:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11794:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11795:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11796:    +' wNumAudioAllocated : WORD; end
11797:   //PCaptureParms', '^TCaptureParms // will not work
11798:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11799:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11800:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11801:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11802:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11803:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11804:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11805:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11806:    +'he : BOOL; AVStreamMaster : WORD; end
11807:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11808:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11809:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11810:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11811:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
       : Integer; hwndParent : THandle; nID : Integer) : THandle
```

```
11812:  Function capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11813:  'IDS_CAP_BEGIN','LongInt'( 300);
11814:  'IDS_CAP_END','LongInt'( 301);
11815:  'IDS_CAP_INFO','LongInt'( 401);
11816:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11817:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11818:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11819:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11820:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11821:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11822:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11823:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11824:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11825:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11826:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11827:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11828:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11829:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11830:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11831:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11832:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11833:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11834:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11835:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11836:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11837:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11838:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11839:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11840:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11841:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11842:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11843:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11844:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11845:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11846:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11847:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11848:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11849:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11850:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11851:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11852:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11853:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11854:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11855:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11856:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11857:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11858:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11859:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11860:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11861:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11862:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11863:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11864:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11865:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11866:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11867:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11868:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11869:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11870:  'AVICAP32','String 'AVICAP32.dll
11871: end;
11872:
11873: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11874: begin
11875:  Function AlBoolToInt( Value : Boolean) : Integer
11876:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11877:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11878:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11879:  Function ALInc( var x : integer; Count : integer) : Integer
11880:  Function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11881:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11882:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11883:  Function ALIsInteger(const S: AnsiString): Boolean;
11884:  function ALIsDecimal(const S: AnsiString): boolean;
11885:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11886:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11887:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11888:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11889:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11890:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11891:  Function ALRandomStr(const aLength: Longint): AnsiString;
11892:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11893:  Function ALRandomStrU(const aLength: Longint): String;
11894: end;
11895:
11896: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11897: begin
11898:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const aTrueStr: AnsiString; const aFalseStr : AnsiString)
```

```
11899: end;
11900:
11901: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11902: begin
11903:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11904:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11905:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11906:    +'; ullAvailExtendedVirtual : Int64; end
11907:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11908:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11909:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11910:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11911:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11912:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11913:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11914:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11915: end;
11916:
11917: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11918: begin
11919:   SIRegister_THandledObject(CL);
11920:   SIRegister_TEvent(CL);
11921:   SIRegister_TMutex(CL);
11922:   SIRegister_TSharedMem(CL);
11923:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11924:  'TRACE_BUFFER','String 'TRACE_BUFFER
11925:  'TRACE_MUTEX','String 'TRACE_MUTEX
11926:   //PTraceEntry', '^TTraceEntry // will not work
11927:   SIRegister_TIPCTracer(CL);
11928:  'MAX_CLIENTS','LongInt'( 6);
11929:  'IPCTIMEOUT','LongInt'( 2000);
11930:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11931:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11932:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11933:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11934:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11935:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11936:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11937:   FindClass('TOBJECT'),'EMonitorActive
11938:   FindClass('TOBJECT'),'TIPCThread
11939:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11940:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11941:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11942:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11943:   TClientFlags', 'set of TClientFlag
11944:   //PEventData', '^TEventData // will not work
11945:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11946:    +'lag; Flags : TClientFlags; end
11947:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11948:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11949:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11950:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11951:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11952:   SIRegister_TIPCEvent(CL);
11953:   //PClientDirRecords', '^TClientDirRecords // will not work
11954:   SIRegister_TClientDirectory(CL);
11955:   TIPCState', '( stInActive, stDisconnected, stConnected )
11956:   SIRegister_TIPCThread(CL);
11957:   SIRegister_TIPCMonitor(CL);
11958:   SIRegister_TIPCClient(CL);
11959:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11960: end;
11961:
11962: (*----------------------------------------------------------------------------*)
11963: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11964: begin
11965:   SIRegister_TAlGSMComm(CL);
11966:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11967:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
      AMessage:AnsiString);
11968:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11969:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
      UseGreekAlphabet:Bool):Widestring;
11970:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11971:  end;
11972:
11973: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11974: begin
11975:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11976:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11977:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11978:   TInternetScheme', 'integer
11979:   TALIPv6Binary', 'array[1..16] of Char;
11980:  // TALIPv6Binary = array[1..16] of ansiChar;
11981:  //   TInternetScheme = Integer;
11982:   SIRegister_TALHTTPRequestHeader(CL);
11983:   SIRegister_TALHTTPCookie(CL);
11984:   SIRegister_TALHTTPCookieCollection(CL);
11985:   SIRegister_TALHTTPResponseHeader(CL);
```

```
11986:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11987:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11988: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
       Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11989: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
       Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11990: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11991:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11992:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11993:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11994:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11995:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11996:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
       ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11997:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
       Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11998:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11999:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
12000:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
12001:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
12002:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
12003:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
12004:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
12005:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
12006:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
12007:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
12008:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
12009:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
12010:  Function ALZeroIpV6 : TALIPv6Binary
12011:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
12012:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
12013:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
12014:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
12015: end;
12016:
12017: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
12018: begin
12019:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
12020:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
12021:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
12022:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
12023:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
12024:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
12025:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
12026:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
12027:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
12028:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12029:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
12030: end;
12031:
12032: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12033: begin
12034:   SIRegister_TALEMailHeader(CL);
12035:   SIRegister_TALNewsArticleHeader(CL);
12036:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
12037:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
12038:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
12039:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
12040:  Function AlGenerateInternetMessageID : AnsiString;
12041:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
12042:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
12043:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12044: end;
12045:
12046: (*----------------------------------------------------------------------------*)
12047: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12048: begin
12049:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12050:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
12051:  Function ALgetLocalIPs : TALStrings
12052:  Function ALgetLocalHostName : AnsiString
12053: end;
12054:
12055: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12056: begin
12057:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
12058:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12059:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12060:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
12061:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
```

```
12062:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12063:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
12064:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12065:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
12066: end;
12067:
12068: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12069: begin
12070:   TStartupInfoA', 'TStartupInfo
12071:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12072:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12073:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12074:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12075:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12076:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12077:  SE_TCB_NAME','String 'SeTcbPrivilege
12078:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12079:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12080:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12081:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12082:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12083:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12084:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12085:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12086:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12087:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12088:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12089:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12090:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12091:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12092:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12093:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12094:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12095:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12096:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12097:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12098:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12099:  Function AlGetEnvironmentString : AnsiString
12100:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12101:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12102:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12103:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12104:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12105: end;
12106:
12107: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12108: begin
12109:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12110:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12111:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12112:  Function ALGetModuleName : ansistring
12113:  Function ALGetModuleFileNameWithoutExtension : ansistring
12114:  Function ALGetModulePath : ansiString
12115:  Function AlGetFileSize( const AFileName : ansistring) : int64
12116:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12117:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12118:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12119:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12120:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12121:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12122:  Function ALFileExists( const Path : ansiString) : boolean
12123:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12124:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12125:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12126:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12127:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12128: end;
12129:
12130: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12131: begin
12132:   NativeInt', 'Integer
12133:   NativeUInt', 'Cardinal
12134:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12135:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12136:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12137:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12138:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12139:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12140:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
```

```
12141:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
        InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12142:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
        InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12143:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
        InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12144:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
        InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12145:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12146:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
        ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12147:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
        OutputBuf:TByteDynArray);
12148:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
        OutputBuffer:TByteDynArray);
12149:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
        OutputBuffer:TByteDynArray);
12150:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
        OutputBuffer:TByteDynArray):NativeInt;
12151:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
        OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12152:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
        ByteBufferSpace:Cardinal):NativeInt;
12153:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12154:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12155:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12156:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12157:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12158:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12159:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12160:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12161:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12162:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12163:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12164:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12165:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12166: end;
12167:
12168: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12169: begin
12170:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12171:   FindClass('TOBJECT'),'TALXMLNode
12172:   FindClass('TOBJECT'),'TALXMLNodeList
12173:   FindClass('TOBJECT'),'TALXMLDocument
12174:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12175:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12176:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12177:    +'nst Name : AnsiString; const Attributes : TALStrings)
12178:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12179:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12180:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12181:    +'ntDocType, ntDocFragment, ntNotation )
12182:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12183:   TALXMLDocOptions', 'set of TALXMLDocOption
12184:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12185:   TALXMLParseOptions', 'set of TALXMLParseOption
12186:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12187:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12188:   SIRegister_EALXMLDocError(CL);
12189:   SIRegister_TALXMLNodeList(CL);
12190:   SIRegister_TALXMLNode(CL);
12191:   SIRegister_TALXmlElementNode(CL);
12192:   SIRegister_TALXmlAttributeNode(CL);
12193:   SIRegister_TALXmlTextNode(CL);
12194:   SIRegister_TALXmlDocumentNode(CL);
12195:   SIRegister_TALXmlCommentNode(CL);
12196:   SIRegister_TALXmlProcessingInstrNode(CL);
12197:   SIRegister_TALXmlCDataNode(CL);
12198:   SIRegister_TALXmlEntityRefNode(CL);
12199:   SIRegister_TALXmlEntityNode(CL);
12200:   SIRegister_TALXmlDocTypeNode(CL);
12201:   SIRegister_TALXmlDocFragmentNode(CL);
12202:   SIRegister_TALXmlNotationNode(CL);
12203:   SIRegister_TALXMLDocument(CL);
12204:  cAlXMLUTF8EncodingStr','String 'UTF-8
12205:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12206:  CALNSDelim','String ':
12207:  CALXML','String 'xml
12208:  CALVersion','String 'version
12209:  CALEncoding','String 'encoding
12210:  CALStandalone','String 'standalone
12211:  CALDefaultNodeIndent','String '
12212:  CALXmlDocument','String 'DOCUMENT
12213:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12214:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
        EncodingStr:AnsiString);
12215:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
        ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12216:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
        ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
```

```
12217:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
       Recurse: Boolean):TalxmlNode
12218:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
       AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12219:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
       AnsiString
12220: end;
12221:
12222: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12223: //based on TEEProc, TeCanvas, TEEngine, TChart
12224: begin
12225:  'TeePiStep','Double').setExtended( Pi / 180.0);
12226:  'TeeDefaultPerspective','LongInt'( 100);
12227:  'TeeMinAngle','LongInt'( 270);
12228:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12229:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12230:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12231:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12232:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12233:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12234:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12235:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12236:  'TA_LEFT','LongInt'( 0);
12237:  'TA_RIGHT','LongInt'( 2);
12238:  'TA_CENTER','LongInt'( 6);
12239:  'TA_TOP','LongInt'( 0);
12240:  'TA_BOTTOM','LongInt'( 8);
12241:  'teePATCOPY','LongInt'( 0);
12242:  'NumCirclePoints','LongInt'( 64);
12243:  'teeDEFAULT_CHARSET','LongInt'( 1);
12244:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12245:  'TA_LEFT','LongInt'( 0);
12246:  'bs_Solid','LongInt'( 0);
12247:  'teepf24Bit','LongInt'( 0);
12248:  'teepfDevice','LongInt'( 1);
12249:  'CM_MOUSELEAVE','LongInt'( 10000);
12250:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12251:  'DC_BRUSH','LongInt'( 18);
12252:  'DC_PEN','LongInt'( 19);
12253:   teeCOLORREF', 'LongWord
12254:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12255:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12256:   SIRegister_TFilterRegion(CL);
12257:   SIRegister_IFormCreator(CL);
12258:   SIRegister_TTeeFilter(CL);
12259:   //TFilterClass', 'class of TTeeFilter
12260:   SIRegister_TFilterItems(CL);
12261:   SIRegister_TConvolveFilter(CL);
12262:   SIRegister_TBlurFilter(CL);
12263:   SIRegister_TTeePicture(CL);
12264:   TPenEndStyle', '( esRound, esSquare, esFlat )
12265:   SIRegister_TChartPen(CL);
12266:   SIRegister_TChartHiddenPen(CL);
12267:   SIRegister_TDottedGrayPen(CL);
12268:   SIRegister_TDarkGrayPen(CL);
12269:   SIRegister_TWhitePen(CL);
12270:   SIRegister_TChartBrush(CL);
12271:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12272:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12273:   SIRegister_TView3DOptions(CL);
12274:   FindClass('TOBJECT'),'TTeeCanvas
12275:   TTeeTransparency', 'Integer
12276:   SIRegister_TTeeBlend(CL);
12277:   FindClass('TOBJECT'),'TCanvas3D
12278:   SIRegister_TTeeShadow(CL);
12279:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
      gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12280:   FindClass('TOBJECT'),'TSubGradient
12281:   SIRegister_TCustomTeeGradient(CL);
12282:   SIRegister_TSubGradient(CL);
12283:   SIRegister_TTeeGradient(CL);
12284:   SIRegister_TTeeFontGradient(CL);
12285:   SIRegister_TTeeFont(CL);
12286:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12287:   TCanvasTextAlign', 'Integer
12288:   TTeeCanvasHandle', 'HDC
12289:   SIRegister_TTeeCanvas(CL);
12290:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12291:   SIRegister_TFloatXYZ(CL);
12292:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12293:   TRGB', 'record blue: byte; green: byte; red: byte; end
12294:   {TRGB=packed record
12295:     Blue  : Byte;
12296:     Green : Byte;
12297:     Red   : Byte;
12298:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12299:
12300: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12301:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
```

```
12302: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12303: TCanvas3DPlane', '( cpX, cpY, cpZ )
12304:   //IInterface', 'IUnknown
12305:   SIRegister_TCanvas3D(CL);
12306:   SIRegister_TTeeCanvas3D(CL);
12307:   TTrianglePoints', 'Array[0..2] of TPoint;
12308:   TFourPoints', 'Array[0..3] of TPoint;
12309:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12310:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12311:  Function Point3D( const x, y, z : Integer) : TPoint3D
12312:  Procedure SwapDouble( var a, b : Double)
12313:  Procedure SwapInteger( var a, b : Integer)
12314:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12315:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12316:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12317:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12318:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12319:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12320:  Procedure UnClipCanvas( ACanvas : TCanvas)
12321:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12322:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12323:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12324:  'TeeCharForHeight','String 'W
12325:  'DarkerColorQuantity','Byte').SetUInt( 128);
12326:  'DarkColorQuantity','Byte').SetUInt( 64);
12327:   TButtonGetColorProc', 'Function  : TColor
12328:   SIRegister_TTeeButton(CL);
12329:   SIRegister_TButtonColor(CL);
12330:   SIRegister_TComboFlat(CL);
12331:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12332:  Function TeePoint( const aX, aY : Integer) : TPoint
12333:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12334:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12335:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12336:  Function OrientRectangle( const R : TRect) : TRect
12337:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12338:  Function PolygonBounds( const P : array of TPoint) : TRect
12339:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12340:  Function RGBValue( const Color : TColor) : TRGB
12341:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12342:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12343:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12344:  Function TeeCull( const P : TFourPoints) : Boolean;
12345:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12346:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12347:  Procedure SmoothStretch( Src, Dst : TBitmap);
12348:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12349:  Function TeeDistance( const x, y : Double) : Double
12350:  Function TeeLoadLibrary( const FileName : String) : HInst
12351:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12352:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12353:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12354:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
       Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12355:   SIRegister_ICanvasHyperlinks(CL);
12356:   SIRegister_ICanvasToolTips(CL);
12357:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12358: end;
12359:
12360: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12361: begin
12362:   TOvcHdc', 'Integer
12363:   TOvcHWND', 'Cardinal
12364:   TOvcHdc', 'HDC
12365:   TOvcHWND', 'HWND
12366:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12367:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12368:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12369:  Function DefaultEpoch : Integer
12370:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12371:  Procedure FixRealPrim( P : PChar; DC : Char)
12372:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12373:  Function GetLeftButton : Byte
12374:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12375:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12376:  Function GetShiftFlags : Byte
12377:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12378:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12379:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12380:  Function ovIsForegroundTask : Boolean
12381:  Function ovTrimLeft( const S : string) : string
12382:  Function ovTrimRight( const S : string) : string
12383:  Function ovQuotedStr( const S : string) : string
12384:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12385:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12386:  Function PtrDiff( const P1, P2 : PChar) : Word
12387:  Procedure PtrInc( var P, Delta : Word)
12388:  Procedure PtrDec( var P, Delta : Word)
12389:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
```

```
12390:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
        SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12391:  Function ovMinI( X, Y : Integer) : Integer
12392:  Function ovMaxI( X, Y : Integer) : Integer
12393:  Function ovMinL( X, Y : LongInt) : LongInt
12394:  Function ovMaxL( X, Y : LongInt) : LongInt
12395:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12396:  Function PartialCompare( const S1, S2 : string) : Boolean
12397:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12398:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12399:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12400:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
       TransparentColor : TColor)
12401:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
       TransparentColor : TColorRef)
12402:  Function ovWidthOf( const R : TRect) : Integer
12403:  Function ovHeightOf( const R : TRect) : Integer
12404:  Procedure ovDebugOutput( const S : string)
12405:  Function GetArrowWidth( Width, Height : Integer) : Integer
12406:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12407:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12408:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12409:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12410:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12411:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12412:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12413:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12414:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12415:  Function CreateMetaFile( p1 : PChar) : HDC
12416:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12417:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12418:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12419:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12420:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12421:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12422:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12423:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12424:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12425:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12426:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12427:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12428:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
       SrcHeight:Int;Rop:DWORD):BOOL
12429:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12430:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
       SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12431:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12432:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12433:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12434:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12435:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12436:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12437:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12438:  Function UpdateColors( DC : HDC) : BOOL
12439:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12440:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12441:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12442:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12443:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12444:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12445:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
       HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12446:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12447:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12448:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12449:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12450:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12451:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12452:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12453:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12454:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12455:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12456:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12457:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12458:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12459: end;
12460:
12461: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12462: begin
12463:   SIRegister_TOvcAbstractStore(CL);
12464:   //PExPropInfo', '^TExPropInfo // will not work
12465: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12466:   SIRegister_TOvcPropertyList(CL);
12467:   SIRegister_TOvcDataFiler(CL);
12468:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12469:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12470:  Function CreateStoredItem( const CompName, PropName : string) : string
12471:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
```

```
12472:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12473: end;
12474:
12475: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12476: begin
12477:  'ovsetsize','LongInt'( 16);
12478:  'etSyntax','LongInt'( 0);
12479:  'etSymantic','LongInt'( 1);
12480:  'chCR','Char #13);
12481:  'chLF','Char #10);
12482:  'chLineSeparator',' chCR);
12483:   SIRegister_TCocoError(CL);
12484:   SIRegister_TCommentItem(CL);
12485:   SIRegister_TCommentList(CL);
12486:   SIRegister_TSymbolPosition(CL);
12487: TGenListType', '( glNever, glAlways, glOnError )
12488: TovBitSet', 'set of Integer
12489:   //PStartTable', '^TStartTable // will not work
12490: 'TovCharSet', 'set of AnsiChar
12491: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12492: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12493: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12494: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12495: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12496:    +'osition; const Data : string; ErrorType : integer)
12497: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12498: TGetCH', 'Function ( pos : longint) : char
12499: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12500:   SIRegister_TCocoRScanner(CL);
12501:   SIRegister_TCocoRGrammar(CL);
12502:  '_EF','Char #0);
12503:  '_TAB','Char').SetString( #09);
12504:  '_CR','Char').SetString( #13);
12505:  '_LF','Char').SetString( #10);
12506:  '_EL','').SetString( _CR);
12507:  '_EOF','Char').SetString( #26);
12508:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12509:  'minErrDist','LongInt'( 2);
12510:  Function ovPadL( S : string; ch : char; L : integer) : string
12511: end;
12512:
12513:   TFormatSettings = record
12514:     CurrencyFormat: Byte;
12515:     NegCurrFormat: Byte;
12516:     ThousandSeparator: Char;
12517:     DecimalSeparator: Char;
12518:     CurrencyDecimals: Byte;
12519:     DateSeparator: Char;
12520:     TimeSeparator: Char;
12521:     ListSeparator: Char;
12522:     CurrencyString: string;
12523:     ShortDateFormat: string;
12524:     LongDateFormat: string;
12525:     TimeAMString: string;
12526:     TimePMString: string;
12527:     ShortTimeFormat: string;
12528:     LongTimeFormat: string;
12529:     ShortMonthNames: array[1..12] of string;
12530:     LongMonthNames: array[1..12] of string;
12531:     ShortDayNames: array[1..7] of string;
12532:     LongDayNames: array[1..7] of string;
12533:     TwoDigitYearCenturyWindow: Word;
12534:   end;
12535:
12536: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12537: begin
12538:  Function ovFormatSettings : TFormatSettings
12539: end;
12540:
12541: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12542: begin
12543:   TOvcCharSet', 'set of Char
12544:   ovBTable', 'array[0..255] of Byte
12545:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
      HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12546:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12547:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12548:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12549:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12550:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12551:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12552:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12553:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12554:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12555:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12556:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12557:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12558:  Function LoCaseChar( C : Char) : Char
12559:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
```

```
12560:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12561:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12562:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12563:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12564:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12565:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12566:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12567:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12568:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12569:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12570:  Procedure TrimAllSpacesPChar( P : PChar)
12571:  Function TrimEmbeddedZeros( const S : string) : string
12572:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12573:  Function TrimTrailPrimPChar( S : PChar) : PChar
12574:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12575:  Function TrimTrailingZeros( const S : string) : string
12576:  Procedure TrimTrailingZerosPChar( P : PChar)
12577:  Function UpCaseChar( C : Char) : Char
12578:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12579:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12580:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12581: end;
12582:
12583: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12584: begin
12585:   //PRaiseFrame', '^TRaiseFrame // will not work
12586:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12587:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12588:  Procedure SafeCloseHandle( var Handle : THandle)
12589:  Procedure ExchangeInteger( X1, X2 : Integer)
12590:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12591:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12592:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12593:
12594: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12595:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12596: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12597:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12598:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12599:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12600:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12601:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12602:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12603:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12604:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12605:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12606:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12607:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12608:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12609:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12610:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12611:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12612:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12613:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12614:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12615:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12616:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12617:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12618:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12619:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12620:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12621:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12622:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12623:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12624:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12625:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12626:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12627:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12628:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12629:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12630:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12631:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12632:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12633:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12634:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12635:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12636:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12637:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12638:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12639:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12640:       var lpLuid: TLargeInteger): BOOL; stdcall;
12641:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12642:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12643:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12644:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12645:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12646:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12647:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12648:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
```

```
12649:       var GenerateOnClose: BOOL): BOOL; stdcall;
12650:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12651:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12652:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12653:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12654:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12655:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12656:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12657:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12658:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12659:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12660:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12661:       var phkResult: HKEY): Longint; stdcall;
12662:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12663:       var phkResult: HKEY): Longint; stdcall;
12664:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12665:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12666:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12667:       lpdwDisposition: PDWORD): Longint; stdcall;
12668:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12669:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12670:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12671:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12672:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12673:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12674:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12675:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12676:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12677:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12678:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12679:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12680:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12681:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12682:       lpcbClass: PDWORD; lpReserved: Pointer;
12683:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12684:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12685:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12686:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12687:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12688:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12689:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12690:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12691:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12692:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12693:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12694:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12695:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12696:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12697:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12698:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12699:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12700:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12701:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12702:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12703:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12704:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12705:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12706:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12707:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12708:
12709:  Function wAddAtom( lpString : PKOLChar) : ATOM
12710:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12711:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12712:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12713:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
      lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12714:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12715:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12716:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12717:  Function wCreateDirectoryEx(lpTemplateDirectory,
      lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12718:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
      bInitialState:BOOL;lpName:PKOLChar):THandle;
12719:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
      PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12720:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
      dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12721:  Function wCreateHardLink(lpFileName,
      lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12722:  Function
      CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12723:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
      nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12724:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
      lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
      Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
      lpProcessInfo:TProcessInformation):BOOL
```

```
12725:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
        Longint; lpName : PKOLChar) : THandle
12726:  Function
        wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12727:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12728:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12729:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12730:  //Function
        wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12731:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12732:  //Function
        wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12733:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12734:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12735:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12736:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12737:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12738:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12739:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
        dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12740:  Function wFindAtom( lpString : PKOLChar) : ATOM
12741:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12742:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12743:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12744:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12745:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12746:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12747:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12748:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12749:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12750:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12751:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12752:  Function wGetCommandLine : PKOLChar
12753:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12754:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12755:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12756:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12757:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12758:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12759:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12760:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12761:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12762:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12763:  Function wGetEnvironmentStrings : PKOLChar
12764:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12765:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12766:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12767:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12768:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12769:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12770:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12771:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12772:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12773:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12774:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12775:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12776:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12777:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
        nSize:DWORD; lpFileName : PKOLChar) : DWORD
12778:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12779:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12780:  Function wGetProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12781:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12782:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12783: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
        lpCharType):BOOL
12784:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12785:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12786:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12787: //Function
        wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12788: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12789: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
        : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
        lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
```

```
12790:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12791:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12792:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12793:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12794:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12795:  Function
       wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12796:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12797:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12798:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12799:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12800:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
       TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12801:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12802:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12803:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12804:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12805:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12806:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12807:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
       lpNumberOfEventsRead:DWORD):BOOL;
12808:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12809:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12810:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
       lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12811:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
       lpNumbOfEventsRead:DWORD):BOOL;
12812:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
       : TCoord; var lpReadRegion : TSmallRect) : BOOL
12813:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
       DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12814:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12815:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
       lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12816:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
       lpFilePart:PKOLChar):DWORD;
12817:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12818:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12819:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12820:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12821:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12822:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12823:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12824:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12825:  //Function wUpdateResource(hUpdate:THandle;lpType,
       lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12826:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12827:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12828:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
       DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12829:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
       var lpNumberOfEventsWritten : DWORD) : BOOL
12830:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
       TCoord; var lpWriteRegion : TSmallRect) : BOOL
12831:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
       DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12832:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12833:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12834:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12835:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12836:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12837:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12838:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12839:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12840:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12841:  Function wlstrlen( lpString : PKOLChar) : Integer
12842:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
       PNetConnectInfoStruct) : DWORD
12843:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
       lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12844:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
       lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12845:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12846:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12847:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12848:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12849:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12850:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12851:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12852:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
       : PKOLChar; nNameBufSize : DWORD) : DWORD
12853:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12854:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12855:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12856: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
       lpBufferSize:DWORD):DWORD;
12857:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12858: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12859: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
```

```
12860:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12861:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12862:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12863:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12864:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12865: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12866: //Func wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12867: //Func wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12868:  Function wAddFontResource( FileName : PKOLChar) : Integer
12869:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12870:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12871:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12872:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12873:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12874: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12875:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
        fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
        fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12876:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12877:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12878: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12879:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12880:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12881: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
        pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12882: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12883:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12884:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12885:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12886:  //Function wExtTextOut(DC:HDC;X,
        Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12887:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12888:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12889:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12890:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12891: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12892: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12893:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12894:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12895: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12896: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
        lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12897:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12898: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12899:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12900: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12901:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12902:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12903:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12904:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12905:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12906:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12907:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12908:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12909:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12910:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12911:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12912:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12913:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12914:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12915:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12916:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12917:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12918:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12919:  //Function
        wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12920:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12921: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
        dwFlags : DWORD; lParam : Pointer) : Longint
12922:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12923:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12924:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12925:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12926:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12927:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12928: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12929:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12930:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12931:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12932:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12933:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12934:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12935:  //Function wCreateDesktop(lpszDesktop,
        lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
```

```
12936:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
        HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12937:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
        lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12938:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
        hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12939: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
        nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12940:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
        dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12941:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12942:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12943:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12944:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12945:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
        : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12946:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
        : TFNDlgProc; dwInitParam : LPARAM) : Integer
12947:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12948:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
        nIDStaticPath:Integer;uFileType:UINT):Integer;
12949:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
        nIDStaticPath:Int;uFiletype:UINT):Int;
12950:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12951:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12952: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
        cy:Int;Flags:UINT):BOOL;
12953:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12954:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12955:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12956:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
        pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12957: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12958:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12959:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12960:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12961:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12962:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12963:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12964:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12965:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12966:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12967:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12968:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12969:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
        lpnTabStopPositions) : DWORD
12970: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
        lpnLengthNeed:DWORD)BOOL;
12971:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12972:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12973:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12974:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12975: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
        nHeigt:Int):BOOL;
12976:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12977:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12978:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12979:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12980:  Function wIsCharLower( ch : KOLChar) : BOOL
12981:  Function wIsCharUpper( ch : KOLChar) : BOOL
12982:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12983:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12984:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12985:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12986:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12987:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12988:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12989:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12990:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12991:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12992:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12993:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12994:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
12995:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
12996:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12997:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
12998:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12999:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
13000:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
13001:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
13002:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
13003:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
13004:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
13005:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
13006:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13007:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13008:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
13009: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
13010: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
```

```
13011:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
13012: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
13013:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
13014:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
13015:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13016:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
13017:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
        lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
13018:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
        lpdwResult:DWORD): LRESULT
13019:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13020:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
13021:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
13022:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
13023:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
13024: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
13025:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
13026:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
13027:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
13028:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13029: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
13030:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
        lpnTabStopPositions,nTabOrigin:Int):Longint;
13031:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
13032:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
13033:  Function wVkKeyScan( ch : KOLChar) : SHORT
13034:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13035:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
13036:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
13037:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
13038:
13039: //TestDrive!
13040: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13041:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13042:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
13043:  Function GetLocalUserSidStr( const UserName : string) : string
13044:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
13045:  Function Impersonate2User( const domain : string; const user : string) : boolean
13046:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
13047:  Function KillProcessbyname( const exename : string; var found : integer) : integer
13048:  Function getWinProcessList : TStringList
13049:  function WaitTilClose(hWnd: Integer): Integer;
13050:  function DoUserMsgs: Boolean;
13051:  function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13052:  procedure ShowMsg(hParent: Integer; const Mess, Title: String);  //modal but NOT blockable
13053:  procedure DeleteMsgForm(Handle: Integer);
13054:  procedure DisableForms;
13055:  function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13056: end;
13057:
13058: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13059: begin
13060:  'AfMaxSyncSlots','LongInt'( 64);
13061:  'AfSynchronizeTimeout','LongInt'( 2000);
13062:   TAfSyncSlotID', 'DWORD
13063:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13064:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
13065:   TAfSafeDirectSyncEvent', 'Procedure
13066:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13067:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
13068:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
13069:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
13070:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
13071:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
13072:  Function AfIsSyncMethod : Boolean
13073:  Function AfSyncWnd : HWnd
13074:  Function AfSyncStatistics : TAfSyncStatistics
13075:  Procedure AfClearSyncStatistics
13076: end;
13077:
13078: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13079: begin
13080:  'fBinary','LongWord')( $00000001);
13081:  'fParity','LongWord')( $00000002);
13082:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13083:  'fOutxDsrFlow','LongWord')( $00000008);
13084:  'fDtrControl','LongWord')( $00000030);
13085:  'fDtrControlDisable','LongWord')( $00000000);
13086:  'fDtrControlEnable','LongWord')( $00000010);
13087:  'fDtrControlHandshake','LongWord')( $00000020);
13088:  'fDsrSensitivity','LongWord')( $00000040);
13089:  'fTXContinueOnXoff','LongWord')( $00000080);
13090:  'fOutX','LongWord')( $00000100);
13091:  'fInX','LongWord')( $00000200);
13092:  'fErrorChar','LongWord')( $00000400);
13093:  'fNull','LongWord')( $00000800);
13094:  'fRtsControl','LongWord')( $00003000);
13095:  'fRtsControlDisable','LongWord')( $00000000);
13096:  'fRtsControlEnable','LongWord')( $00001000);
```

```
13097:  'fRtsControlHandshake','LongWord')( $00002000);
13098:  'fRtsControlToggle','LongWord')( $00003000);
13099:  'fAbortOnError','LongWord')( $00004000);
13100:  'fDummy2','LongWord')( $FFFF8000);
13101:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13102:   FindClass('TOBJECT'),'EAfComPortCoreError
13103:   FindClass('TOBJECT'),'TAfComPortCore
13104:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13105:    +'tKind : TAfCoreEvent; Data : DWORD)
13106:   SIRegister_TAfComPortCoreThread(CL);
13107:   SIRegister_TAfComPortEventThread(CL);
13108:   SIRegister_TAfComPortWriteThread(CL);
13109:   SIRegister_TAfComPortCore(CL);
13110:  Function FormatDeviceName( PortNumber : Integer) : string
13111: end;
13112:
13113: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13114: begin
13115:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13116:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13117:   SIRegister_TApplicationFileIO(CL);
13118:   TDataFileCapability', '( dfcRead, dfcWrite )
13119:   TDataFileCapabilities', 'set of TDataFileCapability
13120:   SIRegister_TDataFile(CL);
13121:   //TDataFileClass', 'class of TDataFile
13122:  Function ApplicationFileIODefined : Boolean
13123:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13124:  Function FileStreamExists(const fileName: String) : Boolean
13125:  //Procedure Register
13126: end;
13127:
13128: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13129: begin
13130:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13131:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13132:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13133:   TALFBXScale', 'Integer
13134:   FindClass('TOBJECT'),'EALFBXConvertError
13135:   SIRegister_EALFBXError(CL);
13136:   SIRegister_EALFBXException(CL);
13137:   FindClass('TOBJECT'),'EALFBXGFixError
13138:   FindClass('TOBJECT'),'EALFBXDSQLError
13139:   FindClass('TOBJECT'),'EALFBXDynError
13140:   FindClass('TOBJECT'),'EALFBXGBakError
13141:   FindClass('TOBJECT'),'EALFBXGSecError
13142:   FindClass('TOBJECT'),'EALFBXLicenseError
13143:   FindClass('TOBJECT'),'EALFBXGStatError
13144:   //EALFBXExceptionClass', 'class of EALFBXError
13145:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13146:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13147:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13148:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13149:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13150:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13151:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13152:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13153:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13154:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13155:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13156:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13157:   TALFBXTransParams', 'set of TALFBXTransParam
13158:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13159:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13160:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13161:  'cALFBXMaxParamLength','LongInt'( 125);
13162:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13163:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13164:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13165:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13166:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13167:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13168:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13169:   SIRegister_TALFBXSQLDA(CL);
13170:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13171:   SIRegister_TALFBXPoolStream(CL);
13172:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13173:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13174:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13175:   //TALFBXArrayDesc', 'TISCArrayDesc
13176:   //TALFBXBlobDesc', 'TISCBlobDesc
13177:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13178:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13179:   SIRegister_TALFBXSQLResult(CL);
13180:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13181:   SIRegister_TALFBXSQLParams(CL);
13182:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13183:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13184:    +'atementType : TALFBXStatementType; end
13185:   FindClass('TOBJECT'),'TALFBXLibrary
```

```
13186:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13187:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13188:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13189:    //+' Excep : EALFBXExceptionClass)
13190:   SIRegister_TALFBXLibrary(CL);
13191:  'cAlFBXDateOffset','LongInt'( 15018);
13192:  'cALFBXTimeCoeff','LongInt'( 864000000);
13193:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13194:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13195:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13196:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13197:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13198:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13199:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13200:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13201:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13202:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13203:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13204:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13205:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13206:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13207: end;
13208:
13209: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13210: begin
13211:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13212:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13213:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13214:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13215:    +'teger; First : Integer; CacheThreshold : Integer; end
13216:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13217:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13218:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13219:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13220:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13221:   SIRegister_TALFBXClient(CL);
13222:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13223:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13224:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13225:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13226:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13227:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13228:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13229:   SIRegister_TALFBXConnectionPoolClient(CL);
13230:   SIRegister_TALFBXEventThread(CL);
13231:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13232: end;
13233:
13234: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13235: begin
13236: _OSVERSIONINFOA = record
13237:     dwOSVersionInfoSize: DWORD;
13238:     dwMajorVersion: DWORD;
13239:     dwMinorVersion: DWORD;
13240:     dwBuildNumber: DWORD;
13241:     dwPlatformId: DWORD;
13242:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13243:   end;
13244:  TOSVersionInfoA', '_OSVERSIONINFOA
13245:   TOSVersionInfo', 'TOSVersionInfoA
13246:  'WS_EX_RIGHT','LongWord')( $00001000);
13247:  'WS_EX_LEFT','LongWord')( $00000000);
13248:  'WS_EX_RTLREADING','LongWord')( $00002000);
13249:  'WS_EX_LTRREADING','LongWord')( $00000000);
13250:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13251:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13252:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13253:  'LAYOUT_RTL','LongWord')( $00000001);
13254:  'LAYOUT_BTT','LongWord')( $00000002);
13255:  'LAYOUT_VBH','LongWord')( $00000004);
13256:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13257:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13258:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13259:  Function GetLayout( dc : hdc) : DWORD
13260:  Function IsBidi : Boolean
13261:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13262:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13263:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13264:  Function GetPriorityClass( hProcess : THandle) : DWORD
13265:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13266:  Function CloseClipboard : BOOL
13267:  Function GetClipboardSequenceNumber : DWORD
13268:  Function GetClipboardOwner : HWND
13269:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13270:  Function GetClipboardViewer : HWND
13271:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13272:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13273:  Function GetClipboardData( uFormat : UINT) : THandle
13274:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
```

```
13275:  Function CountClipboardFormats : Integer
13276:  Function EnumClipboardFormats( format : UINT) : UINT
13277:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13278:  Function EmptyClipboard : BOOL
13279:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13280:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13281:  Function GetOpenClipboardWindow : HWND
13282:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13283:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13284:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13285:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13286:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13287:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13288:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13289:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13290:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13291: end;
13292:
13293: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13294: begin
13295:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13296:  Function GetTemporaryFilesPath : String
13297:  Function GetTemporaryFileName : String
13298:  Function FindFileInPaths( const fileName, paths : String) : String
13299:  Function PathsToString( const paths : TStrings) : String
13300:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13301:  //Function MacroExpandPath( const aPath : String) : String
13302: end;
13303:
13304: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13305: begin
13306:   SIRegister_TALMultiPartBaseContent(CL);
13307:   SIRegister_TALMultiPartBaseContents(CL);
13308:   SIRegister_TAlMultiPartBaseStream(CL);
13309:   SIRegister_TALMultipartBaseEncoder(CL);
13310:   SIRegister_TALMultipartBaseDecoder(CL);
13311:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13312:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13313:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13314: end;
13315:
13316: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13317: begin
13318:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13319:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13320:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13321:  Function aAllocPadedMem( Size : Cardinal) : TObject
13322:  Procedure aFreePadedMem( var P : TObject);
13323:  Procedure aFreePadedMem1( var P : PChar);
13324:  Function aCheckPadedMem( P : Pointer) : Byte
13325:  Function aGetPadMemSize( P : Pointer) : Cardinal
13326:  Function aAllocMem( Size : Cardinal) : Pointer
13327:  Function aStrLen( const Str : PChar) : Cardinal
13328:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13329:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13330:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13331:  Function aStrEnd( const Str : PChar) : PChar
13332:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13333:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13334:  Function aPCharLength( const Str : PChar) : Cardinal
13335:  Function aPCharUpper( Str : PChar) : PChar
13336:  Function aPCharLower( Str : PChar) : PChar
13337:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13338:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13339:  Function aCopyTail( const S : String; Len : Integer) : String
13340:  Function aInt2Thos( I : Int64) : String
13341:  Function aUpperCase( const S : String) : String
13342:  Function aLowerCase( const S : string) : String
13343:  Function aCompareText( const S1, S2 : string) : Integer
13344:  Function aSameText( const S1, S2 : string) : Boolean
13345:  Function aInt2Str( Value : Int64) : String
13346:  Function aStr2Int( const Value : String) : Int64
13347:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13348:  Function aGetFileExt( const FileName : String) : String
13349:  Function aGetFilePath( const FileName : String) : String
13350:  Function aGetFileName( const FileName : String) : String
13351:  Function aChangeExt( const FileName, Extension : String) : String
13352:  Function aAdjustLineBreaks( const S : string) : string
13353:  Function aGetWindowStr( WinHandle : HWND) : String
13354:  Function aDiskSpace( Drive : String) : TdriveSize
13355:  Function aFileExists( FileName : String) : Boolean
13356:  Function aFileSize( FileName : String) : Int64
13357:  Function aDirectoryExists( const Name : string) : Boolean
13358:  Function aSysErrorMessage( ErrorCode : Integer) : string
13359:  Function aShortPathName( const LongName : string) : string
13360:  Function aGetWindowVer : TWinVerRec
13361:  procedure InitDriveSpacePtr;
13362: end;
13363:
```

```
13364: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13365: begin
13366:  aZero','LongInt'( 0);
13367:  'makeappDEF','LongInt'( - 1);
13368:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13369:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13370:  'CS_KEYCVTWINDOW','LongInt'( 4);
13371:  'CS_DBLCLKS','LongInt'( 8);
13372:  'CS_OWNDC','LongWord')( $20);
13373:  'CS_CLASSDC','LongWord')( $40);
13374:  'CS_PARENTDC','LongWord')( $80);
13375:  'CS_NOKEYCVT','LongWord')( $100);
13376:  'CS_NOCLOSE','LongWord')( $200);
13377:  'CS_SAVEBITS','LongWord')( $800);
13378:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13379:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13380:  'CS_GLOBALCLASS','LongWord')( $4000);
13381:  'CS_IME','LongWord')( $10000);
13382:  'CS_DROPSHADOW','LongWord')( $20000);
13383:   //PPanelFunc', '^TPanelFunc // will not work
13384:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13385:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13386:   TFontLooks', 'set of TFontLook
13387:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13388:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13389:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13390:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13391:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13392:  Procedure RunMsgLoop( Show : Boolean)
13393:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13394:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
       ID_Number:Cardinal;hFont:Int):Int;
13395:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13396:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13397:  Function MakePanel(Left,Top,Width,Height,
       hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13398:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13399:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13400:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13401: end;
13402:
13403: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13404: begin
13405:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13406:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13407:   TScreenSaverOptions', 'set of TScreenSaverOption
13408:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13409:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13410:   SIRegister_TScreenSaver(CL);
13411:  //Procedure Register
13412:  Procedure SetScreenSaverPassword
13413: end;
13414:
13415: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13416: begin
13417:   FindClass('TOBJECT'),'TXCollection
13418:   SIRegister_EFilerException(CL);
13419:   SIRegister_TXCollectionItem(CL);
13420:   //TXCollectionItemClass', 'class of TXCollectionItem
13421:   SIRegister_TXCollection(CL);
13422:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13423:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13424:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13425:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13426:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13427:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13428: end;
13429:
13430: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13431: begin
13432:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13433:  Procedure xglMapTexCoordToNull
13434:  Procedure xglMapTexCoordToMain
13435:  Procedure xglMapTexCoordToSecond
13436:  Procedure xglMapTexCoordToDual
13437:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13438:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13439:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13440:  Procedure xglBeginUpdate
13441:  Procedure xglEndUpdate
13442:  Procedure xglPushState
13443:  Procedure xglPopState
13444:  Procedure xglForbidSecondTextureUnit
13445:  Procedure xglAllowSecondTextureUnit
13446:  Function xglGetBitWiseMapping : Cardinal
13447: end;
13448:
13449: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
```

```
13450: begin
13451:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13452:   TBaseListOptions', 'set of TBaseListOption
13453:   SIRegister_TBaseList(CL);
13454:   SIRegister_TBaseVectorList(CL);
13455:   SIRegister_TAffineVectorList(CL);
13456:   SIRegister_TVectorList(CL);
13457:   SIRegister_TTexPointList(CL);
13458:   SIRegister_TXIntegerList(CL);
13459:   //PSingleArrayList', '^TSingleArrayList // will not work
13460:   SIRegister_TSingleList(CL);
13461:   SIRegister_TByteList(CL);
13462:   SIRegister_TQuaternionList(CL);
13463:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13464:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13465:  Procedure FastQuickSortLists(startIndex,
       endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13466: end;
13467:
13468: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13469: begin
13470:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13471:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13472:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
       indices:TIntegerList;list:TAffineVectorList);
13473:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
       indices:TIntegerList;list:TAffineVectorList);
13474:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
       normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13475:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13476:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13477:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13478:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13479:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13480:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13481:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13482:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13483:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13484:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
       edgesTriangles : TIntegerList) : TIntegerList
13485:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13486:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
       TPersistentObjectList;
13487:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13488:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
       TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13489: end;
13490:
13491: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13492: begin
13493:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13494:  Procedure FreeMemAndNil( var P : TObject)
13495:  Function PCharOrNil( const S : string) : PChar
13496:   SIRegister_TJclReferenceMemoryStream(CL);
13497:   FindClass('TOBJECT'),'EJclVMTError
13498:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13499:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13500:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13501:   PDynamicIndexList', '^TDynamicIndexList // will not work
13502:   PDynamicAddressList', '^TDynamicAddressList // will not work
13503:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13504:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13505:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13506:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13507:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13508:  Function GetInitTable( AClass : TClass) : PTypeInfo
13509:   PFieldEntry', '^TFieldEntry // will not work}
13510:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13511:  Function JIsClass( Address : Pointer) : Boolean
13512:  Function JIsObject( Address : Pointer) : Boolean
13513:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13514:   TDigitCount', 'Integer
13515:   SIRegister_TJclNumericFormat(CL);
13516:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13517:   TTextHandler', 'Procedure ( const Text : string)
13518: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13519:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13520:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13521:  Function ReadKey : Char   //to and from the DOS console !
13522:   TModuleHandle', 'HINST
13523:   //TModuleHandle', 'Pointer
13524:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13525:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13526:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13527:  Procedure UnloadModule( var Module : TModuleHandle)
13528:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13529:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13530:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
```

```
13531:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13532:   FindClass('TOBJECT'),'EJclConversionError
13533:  Function JStrToBoolean( const S : string) : Boolean
13534:  Function JBooleanToStr( B : Boolean) : string
13535:  Function JIntToBool( I : Integer) : Boolean
13536:  Function JBoolToInt( B : Boolean) : Integer
13537:  'ListSeparator','String ';
13538:  'ListSeparator1','String ':
13539:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13540:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13541:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13542:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13543:  Function ListItemCount( const List, Separator : string) : Integer
13544:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13545:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13546:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13547:  Function SystemTObjectInstance : LongWord
13548:  Function IsCompiledWithPackages : Boolean
13549:  Function JJclGUIDToString( const GUID : TGUID) : string
13550:  Function JJclStringToGUID( const S : string) : TGUID
13551:   SIRegister_TJclIntfCriticalSection(CL);
13552:   SIRegister_TJclSimpleLog(CL);
13553:  Procedure InitSimpleLog( const ALogFileName : string)
13554: end;
13555:
13556: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13557: begin
13558:   FindClass('TOBJECT'),'EJclBorRADException
13559:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13560:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13561:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13562:   TJclBorRADToolPath', 'string
13563:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13564:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13565:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13566:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13567:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13568:  BorRADToolRepositoryFormsPage','String 'Forms
13569:  BorRADToolRepositoryProjectsPage','String 'Projects
13570:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13571:  BorRADToolRepositoryObjectType','String 'Type
13572:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13573:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13574:  BorRADToolRepositoryObjectName','String 'Name
13575:  BorRADToolRepositoryObjectPage','String 'Page
13576:  BorRADToolRepositoryObjectIcon','String 'Icon
13577:  BorRADToolRepositoryObjectDescr','String 'Description
13578:  BorRADToolRepositoryObjectAuthor','String 'Author
13579:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13580:  BorRADToolRepositoryObjectDesigner','String 'Designer
13581:  BorRADToolRepositoryDesignerDfm','String 'dfm
13582:  BorRADToolRepositoryDesignerXfm','String 'xfm
13583:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13584:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13585:  SourceExtensionDelphiPackage','String '.dpk
13586:  SourceExtensionBCBPackage','String '.bpk
13587:  SourceExtensionDelphiProject','String '.dpr
13588:  SourceExtensionBCBProject','String '.bpr
13589:  SourceExtensionBDSProject','String '.bdsproj
13590:  SourceExtensionDProject','String '.dproj
13591:  BinaryExtensionPackage','String '.bpl
13592:  BinaryExtensionLibrary','String '.dll
13593:  BinaryExtensionExecutable','String '.exe
13594:  CompilerExtensionDCP','String '.dcp
13595:  CompilerExtensionBPI','String '.bpi
13596:  CompilerExtensionLIB','String '.lib
13597:  CompilerExtensionTDS','String '.tds
13598:  CompilerExtensionMAP','String '.map
13599:  CompilerExtensionDRC','String '.drc
13600:  CompilerExtensionDEF','String '.def
13601:  SourceExtensionCPP','String '.cpp
13602:  SourceExtensionH','String '.h
13603:  SourceExtensionPAS','String '.pas
13604:  SourceExtensionDFM','String '.dfm
13605:  SourceExtensionXFM','String '.xfm
13606:  SourceDescriptionPAS','String 'Pascal source file
13607:  SourceDescriptionCPP','String 'C++ source file
13608:  DesignerVCL','String 'VCL
13609:  DesignerCLX','String 'CLX
13610:  ProjectTypePackage','String 'package
13611:  ProjectTypeLibrary','String 'library
13612:  ProjectTypeProgram','String 'program
13613:  Personality32Bit','String '32 bit
13614:  Personality64Bit','String '64 bit
13615:  PersonalityDelphi','String 'Delphi
13616:  PersonalityDelphiDotNet','String 'Delphi.net
13617:  PersonalityBCB','String 'C++Builder
13618:  PersonalityCSB','String 'C#Builder
13619:  PersonalityVB','String 'Visual Basic
```

```
13620:  PersonalityDesign','String 'Design
13621:  PersonalityUnknown','String 'Unknown personality
13622:  PersonalityBDS','String 'Borland Developer Studio
13623:  DOFDirectoriesSection','String 'Directories
13624:  DOFUnitOutputDirKey','String 'UnitOutputDir
13625:  DOFSearchPathName','String 'SearchPath
13626:  DOFConditionals','String 'Conditionals
13627:  DOFLinkerSection','String 'Linker
13628:  DOFPackagesKey','String 'Packages
13629:  DOFCompilerSection','String 'Compiler
13630:  DOFPackageNoLinkKey','String 'PackageNoLink
13631:  DOFAdditionalSection','String 'Additional
13632:  DOFOptionsKey','String 'Options
13633:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13634:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13635:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13636:   TJclBorPersonalities', 'set of TJclBorPersonality
13637:   TJclBorDesigner', '( bdVCL, bdCLX )
13638:   TJclBorDesigners', 'set of TJClBorDesigner
13639:   TJclBorPlatform', '( bp32bit, bp64bit )
13640:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13641:   SIRegister_TJclBorRADToolInstallationObject(CL);
13642:   SIRegister_TJclBorlandOpenHelp(CL);
13643:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13644:   TJclHelp2Objects', 'set of TJclHelp2Object
13645:   SIRegister_TJclHelp2Manager(CL);
13646:   SIRegister_TJclBorRADToolIdeTool(CL);
13647:   SIRegister_TJclBorRADToolIdePackages(CL);
13648:   SIRegister_IJclCommandLineTool(CL);
13649:   FindClass('TOBJECT'),'EJclCommandLineToolError
13650:   SIRegister_TJclCommandLineTool(CL);
13651:   SIRegister_TJclBorlandCommandLineTool(CL);
13652:   SIRegister_TJclBCC32(CL);
13653:   SIRegister_TJclDCC32(CL);
13654:   TJclDCC', 'TJclDCC32
13655:   SIRegister_TJclBpr2Mak(CL);
13656:   SIRegister_TJclBorlandMake(CL);
13657:   SIRegister_TJclBorRADToolPalette(CL);
13658:   SIRegister_TJclBorRADToolRepository(CL);
13659:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13660:   TCommandLineTools', 'set of TCommandLineTool
13661:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13662:   SIRegister_TJclBorRADToolInstallation(CL);
13663:   SIRegister_TJclBCBInstallation(CL);
13664:   SIRegister_TJclDelphiInstallation(CL);
13665:   SIRegister_TJclDCCIL(CL);
13666:   SIRegister_TJclBDSInstallation(CL);
13667:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13668:   SIRegister_TJclBorRADToolInstallations(CL);
13669:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13670:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13671:  Function IsDelphiPackage( const FileName : string) : Boolean
13672:  Function IsDelphiProject( const FileName : string) : Boolean
13673:  Function IsBCBPackage( const FileName : string) : Boolean
13674:  Function IsBCBProject( const FileName : string) : Boolean
13675:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13676:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13677:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
       Descript:PString;
13678:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
13679:  function SamePath(const Path1, Path2: string): Boolean;
13680: end;
13681:
13682: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13683: begin
13684:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13685:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13686:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13687:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13688:  'LPathSeparator','String '/
13689:  'LDirDelimiter','String '/
13690:  'LDirSeparator','String ':
13691:  'JXPathDevicePrefix','String '\\.\
13692:  'JXPathSeparator','String '\
13693:  'JXDirDelimiter','String '\
13694:  'JXDirSeparator','String ';
13695:  'JXPathUncPrefix','String '\\
13696:  'faNormalFile','LongWord')( $00000080);
13697:  //'faUnixSpecific',' faSymLink);
13698:   JXTCompactPath', '( cpCenter, cpEnd )
13699:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13700:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13701:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13702:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13703:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13704:
13705:  Function jxPathAddSeparator( const Path : string) : string
13706:  Function jxPathAddExtension( const Path, Extension : string) : string
```

```
13707:  Function jxPathAppend( const Path, Append : string) : string
13708:  Function jxPathBuildRoot( const Drive : Byte) : string
13709:  Function jxPathCanonicalize( const Path : string) : string
13710:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13711:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13712:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13713:  Function jxPathExtractFileDirFixed( const S : string) : string
13714:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13715:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13716:  Function jxPathGetDepth( const Path : string) : Integer
13717:  Function jxPathGetLongName( const Path : string) : string
13718:  Function jxPathGetShortName( const Path : string) : string
13719:  Function jxPathGetLongName( const Path : string) : string
13720:  Function jxPathGetShortName( const Path : string) : string
13721:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13722:  Function jxPathGetTempPath : string
13723:  Function jxPathIsAbsolute( const Path : string) : Boolean
13724:  Function jxPathIsChild( const Path, Base : string) : Boolean
13725:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13726:  Function jxPathIsUNC( const Path : string) : Boolean
13727:  Function jxPathRemoveSeparator( const Path : string) : string
13728:  Function jxPathRemoveExtension( const Path : string) : string
13729:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13730:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13731:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13732:   JxTFileListOptions', 'set of TFileListOption
13733:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13734:   TFileHandler', 'Procedure ( const FileName : string)
13735:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13736:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13737:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
      AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
      FileMatchFunc:TFileMatchFunc):Bool;
13738:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13739:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13740:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13741:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13742:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
      RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13743:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
      IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13744:  Procedure jxCreateEmptyFile( const FileName : string)
13745:  Function jxCloseVolume( var Volume : THandle) : Boolean
13746:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13747:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13748:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13749:  Function jxDelTree( const Path : string) : Boolean
13750:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13751:  Function jxDiskInDrive( Drive : Char) : Boolean
13752:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13753:  Function jxFileCreateTemp( var Prefix : string) : THandle
13754:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13755:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13756:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13757:  Function jxFileExists( const FileName : string) : Boolean
13758:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13759:  Function jxFileRestore( const FileName : string) : Boolean
13760:  Function jxGetBackupFileName( const FileName : string) : string
13761:  Function jxIsBackupFileName( const FileName : string) : Boolean
13762:  Function jxFileGetDisplayName( const FileName : string) : string
13763:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13764:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13765:  Function jxFileGetSize( const FileName : string) : Int64
13766:  Function jxFileGetTempName( const Prefix : string) : string
13767:  Function jxFileGetTypeName( const FileName : string) : string
13768:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13769:  Function jxForceDirectories( Name : string) : Boolean
13770:  Function jxGetDirectorySize( const Path : string) : Int64
13771:  Function jxGetDriveTypeStr( const Drive : Char) : string
13772:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13773:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13774:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13775:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13776:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13777:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
      ResolveSymLinks:Boolean):Integer
13778:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13779:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13780:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13781:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13782:  Function jxGetFileCreation( const FName : string) : TFileTime;
13783:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13784:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13785:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13786:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13787:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13788:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13789:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13790:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
```

```
13791:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13792:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13793:  Function jxGetModulePath( const Module : HMODULE) : string
13794:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13795:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13796:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13797:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13798:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13799:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13800:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13801:  Function jxOpenVolume( const Drive : Char) : THandle
13802:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13803:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13804:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13805:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13806:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13807:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13808:  Procedure jxShredFile( const FileName : string; Times : Integer)
13809:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13810:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13811:  Function jxSymbolicLinkTarget( const Name : string) : string
13812:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13813:   SIRegister_TJclCustomFileAttrMask(CL);
13814:   SIRegister_TJclFileAttributeMask(CL);
13815:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13816:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13817:   TFileSearchOptions', 'set of TFileSearchOption
13818:   TFileSearchTaskID', 'Integer
13819:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13820:    +'hTaskID; const Aborted : Boolean)
13821:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13822:   SIRegister_IJclFileEnumerator(CL);
13823:   SIRegister_TJclFileEnumerator(CL);
13824:  Function JxFileSearch : IJclFileEnumerator
13825:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13826:   JxTFileFlags', 'set of TFileFlag
13827:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13828:   SIRegister_TJclFileVersionInfo(CL);
13829:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13830:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13831:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13832:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13833:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13834:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13835:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13836:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13837:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13838:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13839:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13840:   SIRegister_TJclTempFileStream(CL);
13841:   FindClass('TOBJECT'),'TJclCustomFileMapping
13842:   SIRegister_TJclFileMappingView(CL);
13843:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13844:   SIRegister_TJclCustomFileMapping(CL);
13845:   SIRegister_TJclFileMapping(CL);
13846:   SIRegister_TJclSwapFileMapping(CL);
13847:   SIRegister_TJclFileMappingStream(CL);
13848:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13849:   //PPCharArray', '^TPCharArray // will not work
13850:   SIRegister_TJclMappedTextReader(CL);
13851:   SIRegister_TJclFileMaskComparator(CL);
13852:   FindClass('TOBJECT'),'EJclPathError
13853:   FindClass('TOBJECT'),'EJclFileUtilsError
13854:   FindClass('TOBJECT'),'EJclTempFileStreamError
13855:   FindClass('TOBJECT'),'EJclTempFileStreamError
13856:   FindClass('TOBJECT'),'EJclFileMappingError
13857:   FindClass('TOBJECT'),'EJclFileMappingViewError
13858:  Function jxPathGetLongName2( const Path : string) : string
13859:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13860:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13861:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13862:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13863:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13864:  Procedure jxPathListAddItems( var List : string; const Items : string)
13865:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13866:  Procedure jxPathListDelItems( var List : string; const Items : string)
13867:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13868:  Function jxPathListItemCount( const List : string) : Integer
13869:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13870:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13871:  Function jxPathListItemIndex( const List, Item : string) : Integer
13872:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13873:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13874:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13875:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string) : Integer
```

```
13876: end;
13877:
13878: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13879: begin
13880:  'UTF8FileHeader','String #$ef#$bb#$bf);
13881:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13882:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13883:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13884:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13885:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13886:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13887:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13888:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13889:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13890:  Function lFileIsReadable( const AFilename : string) : boolean
13891:  Function lFileIsWritable( const AFilename : string) : boolean
13892:  Function lFileIsText( const AFilename : string) : boolean
13893:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13894:  Function lFileIsExecutable( const AFilename : string) : boolean
13895:  Function lFileIsSymlink( const AFilename : string) : boolean
13896:  Function lFileIsHardLink( const AFilename : string) : boolean
13897:  Function lFileSize( const Filename : string) : int64;
13898:  Function lGetFileDescription( const AFilename : string) : string
13899:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13900:  Function lTryReadAllLinks( const Filename : string) : string
13901:  Function lDirPathExists( const FileName : String) : Boolean
13902:  Function lForceDirectory( DirectoryName : string) : boolean
13903:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13904:  Function lProgramDirectory : string
13905:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13906:  Function lExtractFileNameOnly( const AFilename : string) : string
13907:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13908:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13909:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13910:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13911:  Function lAppendPathDelim( const Path : string) : string
13912:  Function lChompPathDelim( const Path : string) : string
13913:  Function lTrimFilename( const AFilename : string) : string
13914:  Function lCleanAndExpandFilename( const Filename : string) : string
13915:  Function lCleanAndExpandDirectory( const Filename : string) : string
13916:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13917:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
       AlwaysRequireSharedBaseFolder : Boolean) : string
13918:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13919:  Function lFileIsInPath( const Filename, Path : string) : boolean
13920:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13921:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13922:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13923:  'AllDirectoryEntriesMask','String '*
13924:  Function lGetAllFilesMask : string
13925:  Function lGetExeExt : string
13926:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
       TSearchFileInPathFlags) : string
13927:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
       TSearchFileInPathFlags) : TStrings
13928:  Function lFindDiskFilename( const Filename : string) : string
13929:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13930:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13931:  Function lGetDarwinSystemFilename( Filename : string) : string
13932:   SIRegister_TFileIterator(CL);
13933:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13934:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13935:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13936:   SIRegister_TFileSearcher(CL);
13937:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13938:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13939:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13940:  // TCopyFileFlags', 'set of TCopyFileFlag
13941:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13942:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13943:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13944:  Function lReadFileToString( const Filename : string) : string
13945:  Function lGetTempFilename( const Directory, Prefix : string) : string
13946:  {Function NeedRTLAnsi : boolean
13947:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13948:  Function UTF8ToSys( const s : string) : string
13949:  Function SysToUTF8( const s : string) : string
13950:  Function ConsoleToUTF8( const s : string) : string
13951:  Function UTF8ToConsole( const s : string) : string}
13952:  Function FileExistsUTF8( const Filename : string) : boolean
13953:  Function FileAgeUTF8( const FileName : string) : Longint
13954:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13955:  Function ExpandFileNameUTF8( const FileName : string) : string
13956:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13957:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13958:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13959:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13960:  Procedure FindCloseUTF8( var F : TSearchrec)
13961:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
```

```
13962:  Function FileGetAttrUTF8( const FileName : String) : Longint
13963:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13964:  Function DeleteFileUTF8( const FileName : String) : Boolean
13965:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13966:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13967:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13968:  Function GetCurrentDirUTF8 : String
13969:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13970:  Function CreateDirUTF8( const NewDir : String) : Boolean
13971:  Function RemoveDirUTF8( const Dir : String) : Boolean
13972:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13973:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13974:  Function FileCreateUTF8( const FileName : string) : THandle;
13975:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13976:  Function ParamStrUTF8( Param : Integer) : string
13977:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13978:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13979:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13980:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13981:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13982: end;
13983:
13984: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13985: begin
13986:   //VK_F23 = 134;
13987:   //{$EXTERNALSYM VK_F24}
13988:   //VK_F24 = 135;
13989:  TVirtualKeyCode', 'Integer
13990:  'VK_MOUSEWHEELUP','integer'(134);
13991:  'VK_MOUSEWHEELDOWN','integer'(135);
13992:  Function glIsKeyDown( c : Char) : Boolean;
13993:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13994:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13995:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13996:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13997:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13998:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13999: end;
14000:
14001: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
14002: begin
14003:   TGLPoint', 'TPoint
14004:   //PGLPoint', '^TGLPoint // will not work
14005:   TGLRect', 'TRect
14006:   //PGLRect', '^TGLRect // will not work
14007:   TDelphiColor', 'TColor
14008:   TGLPicture', 'TPicture
14009:   TGLGraphic', 'TGraphic
14010:   TGLBitmap', 'TBitmap
14011:   //TGraphicClass', 'class of TGraphic
14012:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
14013:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14014:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14015:    +'Button; Shift : TShiftState; X, Y : Integer)
14016:   TGLMouseMoveEvent', 'TMouseMoveEvent
14017:   TGLKeyEvent', 'TKeyEvent
14018:   TGLKeyPressEvent', 'TKeyPressEvent
14019:   EGLOSError', 'EWin32Error
14020:   EGLOSError', 'EWin32Error
14021:   EGLOSError', 'EOSError
14022:  'glsAllFilter','string'All // sAllFilter
14023:  Function GLPoint( const x, y : Integer) : TGLPoint
14024:  Function GLRGB( const r, g, b : Byte) : TColor
14025:  Function GLColorToRGB( color : TColor) : TColor
14026:  Function GLGetRValue( rgb : DWORD) : Byte
14027:  Function GLGetGValue( rgb : DWORD) : Byte
14028:  Function GLGetBValue( rgb : DWORD) : Byte
14029:  Procedure GLInitWinColors
14030:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
14031:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
14032:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14033:  Procedure GLInformationDlg( const msg : String)
14034:  Function GLQuestionDlg( const msg : String) : Boolean
14035:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
14036:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
14037:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
14038:  Function GLApplicationTerminated : Boolean
14039:  Procedure GLRaiseLastOSError
14040:  Procedure GLFreeAndNil( var anObject: TObject)
14041:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
14042:  Function GLGetCurrentColorDepth : Integer
14043:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14044:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14045:  Procedure GLSleep( length : Cardinal)
14046:  Procedure GLQueryPerformanceCounter( var val : Int64)
14047:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14048:  Function GLStartPrecisionTimer : Int64
14049:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14050:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
```

```
14051:  Function GLRDTSC : Int64
14052:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14053:  Function GLOKMessageBox( const Text, Caption : string) : Integer
14054:  Procedure GLShowHTMLUrl( Url : String)
14055:  Procedure GLShowCursor( AShow : boolean)
14056:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14057:  Procedure GLGetCursorPos( var point : TGLPoint)
14058:  Function GLGetScreenWidth : integer
14059:  Function GLGetScreenHeight : integer
14060:  Function GLGetTickCount : int64
14061:  function RemoveSpaces(const str : String) : String;
14062:   TNormalMapSpace','( nmsObject, nmsTangent )
14063:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14064:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14065:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
14066:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
14067:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14068: end;
14069:
14070: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14071: begin
14072:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14073:  // PGLStarRecord', '^TGLStarRecord // will not work
14074:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14075:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14076:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14077: end;
14078:
14079:
14080: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14081: begin
14082:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14083:   //PAABB', '^TAABB // will not work
14084:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14085:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14086:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14087:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14088:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14089:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14090:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14091:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14092:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14093:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14094:  Function BBMinX( const c : THmgBoundingBox) : Single
14095:  Function BBMaxX( const c : THmgBoundingBox) : Single
14096:  Function BBMinY( const c : THmgBoundingBox) : Single
14097:  Function BBMaxY( const c : THmgBoundingBox) : Single
14098:  Function BBMinZ( const c : THmgBoundingBox) : Single
14099:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14100:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14101:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14102:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14103:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14104:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14105:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14106:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14107:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14108:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14109:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14110:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14111:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14112:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14113:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14114:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14115:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14116:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14117:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14118:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14119:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14120:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14121:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14122:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14123:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14124:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14125:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14126:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
14127:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14128:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14129:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14130:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14131:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14132:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
       viewportSizeY:Int):TClipRect
14133: end;
14134:
```

```
14135: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14136: begin
14137:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14138:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14139:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14140:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14141:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14142:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14143:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14144:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14145:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14146:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14147:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14148:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14149:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14150:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14151:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14152:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14153:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14154:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14155:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14156:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14157:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14158:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14159:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14160:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14161:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14162: end;
14163:
14164: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14165: begin
14166:  'EPSILON','Single').setExtended( 1e-40);
14167:  'EPSILON2','Single').setExtended( 1e-30);  }
14168:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14169:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14170:  THmgPlane', 'TVector
14171:   TDoubleHmgPlane', 'THomogeneousDblVector
14172:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14173:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14174:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14175:   TSingleArray', 'array of Single
14176:   TTransformations','array [0..15] of Single)
14177:   TPackedRotationMatrix','array [0..2] of Smallint)
14178:   TVertex', 'TAffineVector
14179:   //TVectorGL', 'THomogeneousFltVector
14180:   //TMatrixGL', 'THomogeneousFltMatrix
14181:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14182:  Function glTexPointMake( const s, t : Single) : TTexPoint
14183:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14184:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14185:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14186:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14187:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14188:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14189:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14190:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14191:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14192:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14193:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14194:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14195:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14196:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14197:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14198:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14199:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14200:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14201:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14202:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14203:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14204:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14205:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14206:  Procedure glRstVector( var v : TAffineVector);
14207:  Procedure glRstVector1( var v : TVectorGL);
14208:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14209:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14210:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14211:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14212:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14213:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14214:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14215:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14216:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14217:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14218:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14219:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14220:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Int;dest:PTexPointArray);
14221:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
```

```
14222:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
        PAffineVectorArray);
14223:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14224:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14225:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14226:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14227:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14228:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14229:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14230:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14231:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14232:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14233:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14234:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14235:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14236:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14237:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14238:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14239:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14240:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14241:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14242:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14243:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14244:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14245:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14246:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14247:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14248:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14249:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14250:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14251:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14252:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14253:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14254:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14255:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14256:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14257:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14258:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14259:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14260:  Function glLerp( const start, stop, t : Single) : Single
14261:  Function glAngleLerp( start, stop, t : Single) : Single
14262:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14263:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14264:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14265:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14266:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14267:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14268:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14269:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14270: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14271: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
        PAffineVectorArray);
14272:  Function glVectorLength( const x, y : Single) : Single;
14273:  Function glVectorLength1( const x, y, z : Single) : Single;
14274:  Function glVectorLength2( const v : TAffineVector) : Single;
14275:  Function glVectorLength3( const v : TVectorGL) : Single;
14276:  Function glVectorLength4( const v : array of Single) : Single;
14277:  Function glVectorNorm( const x, y : Single) : Single;
14278:  Function glVectorNorm1( const v : TAffineVector) : Single;
14279:  Function glVectorNorm2( const v : TVectorGL) : Single;
14280:  Function glVectorNorm3( var V : array of Single) : Single;
14281:  Procedure glNormalizeVector( var v : TAffineVector);
14282:  Procedure glNormalizeVector1( var v : TVectorGL);
14283:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14284:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14285: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14286:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14287:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14288:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14289:  Procedure glNegateVector( var V : TAffineVector);
14290:  Procedure glNegateVector2( var V : TVectorGL);
14291:  Procedure glNegateVector3( var V : array of Single);
14292:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14293:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14294:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14295:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14296:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14297:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14298:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14299:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14300:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14301:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14302:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14303:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14304:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14305:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14306:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14307:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14308:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
```

```
14309:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14310:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14311:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14312:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14313:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14314:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14315:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14316:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14317:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14318:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14319:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14320:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14321:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14322:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14323:  Procedure glAbsVector( var v : TVectorGL);
14324:  Procedure glAbsVector1( var v : TAffineVector);
14325:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14326:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14327:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14328:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14329:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14330:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14331:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14332:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14333:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14334:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14335:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14336:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14337:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14338:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14339:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14340:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14341:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14342:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14343:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14344:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14345:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14346:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14347:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14348:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14349:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14350:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14351:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14352:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14353:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14354:  Procedure glAdjointMatrix( var M : TMatrixGL);
14355:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14356:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14357:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14358:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14359:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14360:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14361:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14362:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14363:  Procedure glInvertMatrix( var M : TMatrixGL);
14364:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14365:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14366:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14367:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14368:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14369:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14370:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14371:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14372:  Procedure glNormalizePlane( var plane : THmgPlane)
14373:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14374:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14375:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14376:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14377:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14378:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14379:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14380:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14381:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14382:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14383:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14384:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14385:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14386:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14387:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14388:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14389:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14390:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14391:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14392:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14393:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14394:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14395:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14396:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
```

```
14397:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14398:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14399:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14400:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14401:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14402:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14403:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14404:  Function glLnXP1( X : Extended) : Extended
14405:  Function glLog10( X : Extended) : Extended
14406:  Function glLog2( X : Extended) : Extended;
14407:  Function glLog21( X : Single) : Single;
14408:  Function glLogN( Base, X : Extended) : Extended
14409:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14410:  Function glPower( const Base, Exponent : Single) : Single;
14411:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14412:  Function glDegToRad( const Degrees : Extended) : Extended;
14413:  Function glDegToRad1( const Degrees : Single) : Single;
14414:  Function glRadToDeg( const Radians : Extended) : Extended;
14415:  Function glRadToDeg1( const Radians : Single) : Single;
14416:  Function glNormalizeAngle( angle : Single) : Single
14417:  Function glNormalizeDegAngle( angle : Single) : Single
14418:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14419:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14420:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14421:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14422:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14423:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14424:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14425:  Function glArcCos( const X : Extended) : Extended;
14426:  Function glArcCos1( const x : Single) : Single;
14427:  Function glArcSin( const X : Extended) : Extended;
14428:  Function glArcSin1( const X : Single) : Single;
14429:  Function glArcTan21( const Y, X : Extended) : Extended;
14430:  Function glArcTan21( const Y, X : Single) : Single;
14431:  Function glFastArcTan2( y, x : Single) : Single
14432:  Function glTan( const X : Extended) : Extended;
14433:  Function glTan1( const X : Single) : Single;
14434:  Function glCoTan( const X : Extended) : Extended;
14435:  Function glCoTan1( const X : Single) : Single;
14436:  Function glSinh( const x : Single) : Single;
14437:  Function glSinh1( const x : Double) : Double;
14438:  Function glCosh( const x : Single) : Single;
14439:  Function glCosh1( const x : Double) : Double;
14440:  Function glRSqrt( v : Single) : Single
14441:  Function glRLength( x, y : Single) : Single
14442:  Function glISqrt( i : Integer) : Integer
14443:  Function glILength( x, y : Integer) : Integer;
14444:  Function glILength1( x, y, z : Integer) : Integer;
14445:  Procedure glRegisterBasedExp
14446:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14447:  Function glRoundInt( v : Single) : Single;
14448:  Function glRoundInt1( v : Extended) : Extended;
14449:  Function glTrunc( v : Single) : Integer;
14450:  Function glTrunc64( v : Extended) : Int64;
14451:  Function glInt( v : Single) : Single;
14452:  Function glInt1( v : Extended) : Extended;
14453:  Function glFrac( v : Single) : Single;
14454:  Function glFrac1( v : Extended) : Extended;
14455:  Function glRound( v : Single) : Integer;
14456:  Function glRound64( v : Single) : Int64;
14457:  Function glRound641( v : Extended) : Int64;
14458:  Function glTrunc( X : Extended) : Int64
14459:  Function glRound( X : Extended) : Int64
14460:  Function glFrac( X : Extended) : Extended
14461:  Function glCeil( v : Single) : Integer;
14462:  Function glCeil64( v : Extended) : Int64;
14463:  Function glFloor( v : Single) : Integer;
14464:  Function glFloor64( v : Extended) : Int64;
14465:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14466:  Function glSign( x : Single) : Integer
14467:  Function glIsInRange( const x, a, b : Single) : Boolean;
14468:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14469:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14470:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14471:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14472:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14473:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14474:  Function glMinFloat3( const v1, v2 : Single) : Single;
14475:  Function glMinFloat4( const v : array of Single) : Single;
14476:  Function glMinFloat5( const v1, v2 : Double) : Double;
14477:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14478:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14479:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14480:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14481:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14482:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14483:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14484:  Function glMaxFloat2( const v : array of Single) : Single;
14485:  Function glMaxFloat3( const v1, v2 : Single) : Single;
```

```
14486:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14487:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14488:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14489:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14490:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14491:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14492:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14493:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14494:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14495:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14496:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14497:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14498:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14499:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14500:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14501:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14502:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14503:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14504:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14505:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14506:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14507:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14508:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14509:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14510:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14511:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14512:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14513:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14514:  Procedure glSortArrayAscending( var a : array of Extended)
14515:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14516:  Function glClampValue1( const aValue, aMin : Single) : Single;
14517:  Function glGeometryOptimizationMode : String
14518:  Procedure glBeginFPUOnlySection
14519:  Procedure glEndFPUOnlySection
14520:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14521:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14522:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14523:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14524:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14525:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14526:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14527:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14528:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14529:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14530:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14531:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14532:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14533:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14534:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14535:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14536:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14537:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14538:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14539:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14540:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14541:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14542:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14543:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14544:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14545:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14546:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14547:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14548:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14549:  'cPI','Single').setExtended( 3.141592654);
14550:  'cPIdiv180','Single').setExtended( 0.017453292);
14551:  'c180divPI','Single').setExtended( 57.29577951);
14552:  'c2PI','Single').setExtended( 6.283185307);
14553:  'cPIdiv2','Single').setExtended( 1.570796326);
14554:  'cPIdiv4','Single').setExtended( 0.785398163);
14555:  'c3PIdiv4','Single').setExtended( 2.35619449);
14556:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14557:  'cInv360','Single').setExtended( 1 / 360);
14558:  'c180','Single').setExtended( 180);
14559:  'c360','Single').setExtended( 360);
14560:  'cOneHalf','Single').setExtended( 0.5);
14561:  'cLn10','Single').setExtended( 2.302585093);
14562:  {'MinSingle','Extended').setExtended( 1.5e-45);
14563:  'MaxSingle','Extended').setExtended( 3.4e+38);
14564:  'MinDouble','Extended').setExtended( 5.0e-324);
14565:  'MaxDouble','Extended').setExtended( 1.7e+308);
14566:  'MinExtended','Extended').setExtended( 3.4e-4932);
14567:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14568:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14569:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
```

```
14570: end;
14571:
14572: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14573: begin
14574:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14575:   (FindClass('TOBJECT'),'TFaceGroups
14576:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14577:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14578:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14579:   SIRegister_TBaseMeshObject(CL);
14580:   (FindClass('TOBJECT'),'TSkeletonFrameList
14581:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14582:   SIRegister_TSkeletonFrame(CL);
14583:   SIRegister_TSkeletonFrameList(CL);
14584:   (FindClass('TOBJECT'),'TSkeleton
14585:   (FindClass('TOBJECT'),'TSkeletonBone
14586:   SIRegister_TSkeletonBoneList(CL);
14587:   SIRegister_TSkeletonRootBoneList(CL);
14588:   SIRegister_TSkeletonBone(CL);
14589:   (FindClass('TOBJECT'),'TSkeletonColliderList
14590:   SIRegister_TSkeletonCollider(CL);
14591:   SIRegister_TSkeletonColliderList(CL);
14592:   (FindClass('TOBJECT'),'TGLBaseMesh
14593:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14594:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14595:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14596:    +'QuaternionList; end
14597:   SIRegister_TSkeleton(CL);
14598:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14599:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14600:   SIRegister_TMeshObject(CL);
14601:   SIRegister_TMeshObjectList(CL);
14602:   //TMeshObjectListClass', 'class of TMeshObjectList
14603:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14604:   SIRegister_TMeshMorphTarget(CL);
14605:   SIRegister_TMeshMorphTargetList(CL);
14606:   SIRegister_TMorphableMeshObject(CL);
14607:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14608:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14609:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14610:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14611:   SIRegister_TSkeletonMeshObject(CL);
14612:   SIRegister_TFaceGroup(CL);
14613:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14614:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14615:   SIRegister_TFGVertexIndexList(CL);
14616:   SIRegister_TFGVertexNormalTexIndexList(CL);
14617:   SIRegister_TFGIndexTexCoordList(CL);
14618:   SIRegister_TFaceGroups(CL);
14619:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14620:   SIRegister_TVectorFile(CL);
14621:   //TVectorFileClass', 'class of TVectorFile
14622:   SIRegister_TGLGLSMVectorFile(CL);
14623:   SIRegister_TGLBaseMesh(CL);
14624:   SIRegister_TGLFreeForm(CL);
14625:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14626:   TGLActorOptions', 'set of TGLActorOption
14627:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14628:   (FindClass('TOBJECT'),'TGLActor
14629:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14630:   SIRegister_TActorAnimation(CL);
14631:   TActorAnimationName', 'String
14632:   SIRegister_TActorAnimations(CL);
14633:   SIRegister_TGLBaseAnimationControler(CL);
14634:   SIRegister_TGLAnimationControler(CL);
14635:   TActorFrameInterpolation', '( afpNone, afpLinear )
14636:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14637:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14638:   SIRegister_TGLActor(CL);
14639:   SIRegister_TVectorFileFormat(CL);
14640:   SIRegister_TVectorFileFormatsList(CL);
14641:   (FindClass('TOBJECT'),'EInvalidVectorFile
14642:  Function GetVectorFileFormats : TVectorFileFormatsList
14643:  Function VectorFileFormatsFilter : String
14644:  Function VectorFileFormatsSaveFilter : String
14645:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14646:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14647:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14648: end;
14649:
14650: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14651: begin
14652:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14653:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14654:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14655:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14656:   SIRegister_TOleStream(CL);
14657:   (FindClass('TOBJECT'),'TConnectionPoints
14658:   TConnectionKind', '( ckSingle, ckMulti )
```

```
14659:   SIRegister_TConnectionPoint(CL);
14660:   SIRegister_TConnectionPoints(CL);
14661:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14662:   (FindClass('TOBJECT'),'TActiveXControlFactory
14663:   SIRegister_TActiveXControl(CL);
14664:   //TActiveXControlClass', 'class of TActiveXControl
14665:   SIRegister_TActiveXControlFactory(CL);
14666:   SIRegister_TActiveFormControl(CL);
14667:   SIRegister_TActiveForm(CL);
14668:   //TActiveFormClass', 'class of TActiveForm
14669:   SIRegister_TActiveFormFactory(CL);
14670:   (FindClass('TOBJECT'),'TPropertyPageImpl
14671:   SIRegister_TPropertyPage(CL);
14672:   //TPropertyPageClass', 'class of TPropertyPage
14673:   SIRegister_TPropertyPageImpl(CL);
14674:   SIRegister_TActiveXPropertyPage(CL);
14675:   SIRegister_TActiveXPropertyPageFactory(CL);
14676:   SIRegister_TCustomAdapter(CL);
14677:   SIRegister_TAdapterNotifier(CL);
14678:   SIRegister_IFontAccess(CL);
14679:   SIRegister_TFontAdapter(CL);
14680:   SIRegister_IPictureAccess(CL);
14681:   SIRegister_TPictureAdapter(CL);
14682:   SIRegister_TOleGraphic(CL);
14683:   SIRegister_TStringsAdapter(CL);
14684:   SIRegister_TReflectorWindow(CL);
14685:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14686:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14687:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14688:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14689:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14690:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14691:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14692:  Function ParkingWindow : HWND
14693: end;
14694:
14695: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14696: begin
14697:  // TIp6Bytes = array [0..15] of Byte;
14698: {:binary form of IPv6 adress (for string conversion routines)}
14699:  // TIp6Words = array [0..7] of Word;
14700:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14701:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14702:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14703:  Function synaIsIP( const Value : string) : Boolean');
14704:  Function synaIsIP6( const Value : string) : Boolean');
14705:  Function synaIPToID( Host : string) : Ansistring');
14706:  Function synaStrToIp6( value : string) : TIp6Bytes');
14707:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14708:  Function synaStrToIp( value : string) : integer');
14709:  Function synaIpToStr( value : integer) : string');
14710:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14711:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14712:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14713:  Function xStrToIP( const Value : String) : TIPAdr');
14714:  Function xIPToStr( const Adresse : TIPAdr) : String');
14715:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14716:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14717: end;
14718:
14719: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14720: begin
14721:  AddTypeS('TSpecials', 'set of Char');
14722:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14723:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14724:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14725:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14726:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14727:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14728:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14729:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14730:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14731:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14732:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14733:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14734:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14735:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14736:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14737:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14738:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
14739:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14740:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14741:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14742:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14743:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14744:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14745:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14746:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14747:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
```

```
14748:  Function synCrc32( const Value : AnsiString) : Integer');
14749:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14750:  Function Crc16( const Value : AnsiString) : Word');
14751:  Function synMD5( const Value : AnsiString) : AnsiString');
14752:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14753:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14754:  Function synSHA1( const Value : AnsiString) : AnsiString');
14755:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14756:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14757:  Function synMD4( const Value : AnsiString) : AnsiString');
14758: end;
14759:
14760: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14761: begin
14762:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14763:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14764:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14765:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14766:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14767:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14768:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14769:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14770:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14771:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14772:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14773:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14774:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14775:    +', CP864, CP865, CP869, CP1125 )');
14776:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14777:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14778:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
       TransformTable : array of Word) : AnsiString');
14779:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
       TransformTable : array of Word; Translit : Boolean) : AnsiString');
14780:  Function GetCurCP : TMimeChar');
14781:  Function GetCurOEMCP : TMimeChar');
14782:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14783:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14784:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14785:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14786:  Function GetBOM( Value : TMimeChar) : AnsiString');
14787:  Function StringToWide( const Value : AnsiString) : WideString');
14788:  Function WideToString( const Value : WideString) : AnsiString');
14789: end;
14790:
14791: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14792: begin
14793:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14794:  Procedure WakeOnLan( MAC, IP : string)');
14795:  Function GetDNS : string');
14796:  Function GetIEProxy( protocol : string) : TProxySetting');
14797:  Function GetLocalIPs : string');
14798: end;
14799:
14800:
14801: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14802: begin
14803:  AddConstantN('synCR','Char #$0d);
14804:  Const('synLF','Char #$0a);
14805:  Const('cSerialChunk','LongInt'( 8192);
14806:  Const('LockfileDirectory','String '/var/lock');
14807:  Const('PortIsClosed','LongInt'( - 1);
14808:  Const('ErrAlreadyOwned','LongInt'( 9991);
14809:  Const('ErrAlreadyInUse','LongInt'( 9992);
14810:  Const('ErrWrongParameter','LongInt'( 9993);
14811:  Const('ErrPortNotOpen','LongInt'( 9994);
14812:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14813:  Const('ErrMaxBuffer','LongInt'( 9996);
14814:  Const('ErrTimeout','LongInt'( 9997);
14815:  Const('ErrNotRead','LongInt'( 9998);
14816:  Const('ErrFrame','LongInt'( 9999);
14817:  Const('ErrOverrun','LongInt'( 10000);
14818:  Const('ErrRxOver','LongInt'( 10001);
14819:  Const('ErrRxParity','LongInt'( 10002);
14820:  Const('ErrTxFull','LongInt'( 10003);
14821:  Const('dcb_Binary','LongWord')( $00000001);
14822:  Const('dcb_ParityCheck','LongWord')( $00000002);
14823:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14824:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14825:  Const('dcb_DtrControlMask','LongWord')( $00000030);
14826:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14827:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14828:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14829:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14830:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14831:  Const('dcb_OutX','LongWord')( $00000100);
14832:  Const('dcb_InX','LongWord')( $00000200);
14833:  Const('dcb_ErrorChar','LongWord')( $00000400);
14834:  Const('dcb_NullStrip','LongWord')( $00000800);
```

```
14835:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14836:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14837:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14838:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14839:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14840:  Const('dcb_AbortOnError','LongWord')( $00004000);
14841:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14842:  Const('synSB1','LongInt'( 0);
14843:  Const('SB1andHalf','LongInt'( 1);
14844:  Const('synSB2','LongInt'( 2);
14845:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14846:  Const('CS7fix','LongWord')( $0000020);
14847:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14848:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14849:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14850:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14851:   //AddTypeS('PDCB', '^TDCB // will not work');
14852:  //Const('MaxRates','LongInt'( 18);
14853:  //Const('MaxRates','LongInt'( 30);
14854:  //Const('MaxRates','LongInt'( 19);
14855:  Const('O_SYNC','LongWord')( $0080);
14856:  Const('synOK','LongInt'( 0);
14857:  Const('synErr','LongInt'( integer ( - 1 ));
14858:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14859:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14860:   SIRegister_ESynaSerError(CL);
14861:   SIRegister_TBlockSerial(CL);
14862:  Function GetSerialPortNames : string');
14863: end;
14864:
14865: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14866: begin
14867:  Const('DLLIconvName','String 'libiconv.so');
14868:  Const('DLLIconvName','String 'iconv.dll');
14869:   AddTypeS('size_t', 'Cardinal');
14870:   AddTypeS('iconv_t', 'Integer');
14871:   //AddTypeS('iconv_t', 'Pointer');
14872:   AddTypeS('argptr', 'iconv_t');
14873:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14874:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14875:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14876:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14877:  Function SynaIconvClose( var cd : iconv_t) : integer');
14878:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14879:  Function IsIconvloaded : Boolean');
14880:  Function InitIconvInterface : Boolean');
14881:  Function DestroyIconvInterface : Boolean');
14882:  Const('ICONV_TRIVIALP','LongInt'( 0);
14883:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14884:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14885:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14886:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14887: end;
14888:
14889: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14890: begin
14891:  Const 'ICMP_ECHO','LongInt'( 8);
14892:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14893:  Const('ICMP_UNREACH','LongInt'( 3);
14894:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14895:  Const('ICMP6_ECHO','LongInt'( 128);
14896:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14897:  Const('ICMP6_UNREACH','LongInt'( 1);
14898:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14899:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14900:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14901:   SIRegister_TPINGSend(CL);
14902:  Function PingHost( const Host : string) : Integer');
14903:  Function TraceRouteHost( const Host : string) : string');
14904: end;
14905:
14906: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14907: begin
14908:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14909:  Const('synASN1_INT','LongWord')( $02);
14910:  Const('synASN1_OCTSTR','LongWord')( $04);
14911:  Const('synASN1_NULL','LongWord')( $05);
14912:  Const('synASN1_OBJID','LongWord')( $06);
14913:  Const('synASN1_ENUM','LongWord')( $0a);
14914:  Const('synASN1_SEQ','LongWord')( $30);
14915:  Const('synASN1_SETOF','LongWord')( $31);
14916:  Const('synASN1_IPADDR','LongWord')( $40);
14917:  Const('synASN1_COUNTER','LongWord')( $41);
14918:  Const('synASN1_GAUGE','LongWord')( $42);
14919:  Const('synASN1_TIMETICKS','LongWord')( $43);
14920:  Const('synASN1_OPAQUE','LongWord')( $44);
14921:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14922:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
```

```
14923:  Function synASNEncLen( Len : Integer) : AnsiString');
14924:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14925:  Function synASNEncInt( Value : Integer) : AnsiString');
14926:  Function synASNEncUInt( Value : Integer) : AnsiString');
14927:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14928:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14929:  Function synMibToId( Mib : String) : AnsiString');
14930:  Function synIdToMib( const Id : AnsiString) : String');
14931:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14932:  Function ASNdump( const Value : AnsiString) : AnsiString');
14933:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14934:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14935: end;
14936:
14937: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14938: begin
14939:  Const('cLDAPProtocol','String '389');
14940:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14941:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14942:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14943:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14944:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14945:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14946:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14947:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14948:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14949:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14950:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14951:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14952:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14953:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14954:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14955:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14956:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14957:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14958:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14959:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14960:   SIRegister_TLDAPAttribute(CL);
14961:   SIRegister_TLDAPAttributeList(CL);
14962:   SIRegister_TLDAPResult(CL);
14963:   SIRegister_TLDAPResultList(CL);
14964:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14965:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14966:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14967:   SIRegister_TLDAPSend(CL);
14968:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14969: end;
14970:
14971:
14972: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14973: begin
14974:  Const('cSysLogProtocol','String '514');
14975:  Const('FCL_Kernel','LongInt'( 0);
14976:  Const('FCL_UserLevel','LongInt'( 1);
14977:  Const('FCL_MailSystem','LongInt'( 2);
14978:  Const('FCL_System','LongInt'( 3);
14979:  Const('FCL_Security','LongInt'( 4);
14980:  Const('FCL_Syslogd','LongInt'( 5);
14981:  Const('FCL_Printer','LongInt'( 6);
14982:  Const('FCL_News','LongInt'( 7);
14983:  Const('FCL_UUCP','LongInt'( 8);
14984:  Const('FCL_Clock','LongInt'( 9);
14985:  Const('FCL_Authorization','LongInt'( 10);
14986:  Const('FCL_FTP','LongInt'( 11);
14987:  Const('FCL_NTP','LongInt'( 12);
14988:  Const('FCL_LogAudit','LongInt'( 13);
14989:  Const('FCL_LogAlert','LongInt'( 14);
14990:  Const('FCL_Time','LongInt'( 15);
14991:  Const('FCL_Local0','LongInt'( 16);
14992:  Const('FCL_Local1','LongInt'( 17);
14993:  Const('FCL_Local2','LongInt'( 18);
14994:  Const('FCL_Local3','LongInt'( 19);
14995:  Const('FCL_Local4','LongInt'( 20);
14996:  Const('FCL_Local5','LongInt'( 21);
14997:  Const('FCL_Local6','LongInt'( 22);
14998:  Const('FCL_Local7','LongInt'( 23);
14999:   Type(TSyslogSeverity','(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
15000:   SIRegister_TSyslogMessage(CL);
15001:   SIRegister_TSyslogSend(CL);
15002:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
      Content:string):Boolean;
15003: end;
15004:
15005:
15006: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
15007: begin
15008:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
15009:   SIRegister_TMessHeader(CL);
15010:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
```

```
15011:   SIRegister_TMimeMess(CL);
15012: end;
15013:
15014: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
15015: begin
15016:   (FindClass('TOBJECT'),'TMimePart');
15017:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15018:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
15019:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15020:   SIRegister_TMimePart(CL);
15021:  Const('MaxMimeType','LongInt'( 25);
15022:  Function GenerateBoundary : string');
15023: end;
15024:
15025: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
15026: begin
15027:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
15028:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
15029:  Function NeedInline( const Value : AnsiString) : boolean');
15030:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15031:  Function InlineCode( const Value : string) : string');
15032:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15033:  Function InlineEmail( const Value : string) : string');
15034: end;
15035:
15036: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15037: begin
15038:  Const('cFtpProtocol','String '21');
15039:  Const('cFtpDataProtocol','String '20');
15040:  Const('FTP_OK','LongInt'( 255);
15041:  Const('FTP_ERR','LongInt'( 254);
15042:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15043:   SIRegister_TFTPListRec(CL);
15044:   SIRegister_TFTPList(CL);
15045:   SIRegister_TFTPSend(CL);
15046:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15047:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15048:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
      ToPort, ToFile, ToUser, ToPass : string) : Boolean');
15049: end;
15050:
15051: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15052: begin
15053:  Const('cHttpProtocol','String '80');
15054:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15055:   SIRegister_THTTPSend(CL);
15056:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
15057:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
15058:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
15059:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
15060:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
      ResultData:TStrings):Bool;
15061: end;
15062:
15063: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15064: begin
15065:  Const('cSmtpProtocol','String '25');
15066:   SIRegister_TSMTPSend(CL);
15067:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
      Passw:string):Bool;
15068:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15069:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
      Username, Password : string):Boolean');
15070: end;
15071:
15072: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15073: begin
15074:  Const('cSnmpProtocol','String '161');
15075:  Const('cSnmpTrapProtocol','String '162');
15076:  Const('SNMP_V1','LongInt'( 0);
15077:  Const('SNMP_V2C','LongInt'( 1);
15078:  Const('SNMP_V3','LongInt'( 3);
15079:  Const('PDUGetRequest','LongWord')( $A0);
15080:  Const('PDUGetNextRequest','LongWord')( $A1);
15081:  Const('PDUGetResponse','LongWord')( $A2);
15082:  Const('PDUSetRequest','LongWord')( $A3);
15083:  Const('PDUTrap','LongWord')( $A4);
15084:  Const('PDUGetBulkRequest','LongWord')( $A5);
15085:  Const('PDUInformRequest','LongWord')( $A6);
15086:  Const('PDUTrapV2','LongWord')( $A7);
15087:  Const('PDUReport','LongWord')( $A8);
15088:  Const('ENoError',LongInt 0;
15089:  Const('ETooBig','LongInt')( 1);
15090:  Const('ENoSuchName','LongInt'( 2);
15091:  Const('EBadValue','LongInt'( 3);
15092:  Const('EReadOnly','LongInt'( 4);
15093:  Const('EGenErr','LongInt'( 5);
15094:  Const('ENoAccess','LongInt'( 6);
15095:  Const('EWrongType','LongInt'( 7);
```

```
15096:  Const('EWrongLength','LongInt'( 8);
15097:  Const('EWrongEncoding','LongInt'( 9);
15098:  Const('EWrongValue','LongInt'( 10);
15099:  Const('ENoCreation','LongInt'( 11);
15100:  Const('EInconsistentValue','LongInt'( 12);
15101:  Const('EResourceUnavailable','LongInt'( 13);
15102:  Const('ECommitFailed','LongInt'( 14);
15103:  Const('EUndoFailed','LongInt'( 15);
15104:  Const('EAuthorizationError','LongInt'( 16);
15105:  Const('ENotWritable','LongInt'( 17);
15106:  Const('EInconsistentName','LongInt'( 18);
15107:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15108:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15109:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15110:   SIRegister_TSNMPMib(CL);
15111:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15112:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15113:   SIRegister_TSNMPRec(CL);
15114:   SIRegister_TSNMPSend(CL);
15115:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15116:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15117:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15118:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15119:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15120:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15121:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : TStringList) : Integer');
15122: end;
15123:
15124: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15125: begin
15126:  Function GetDomainName2: AnsiString');
15127:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15128:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15129:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15130:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15131:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15132: end;
15133:
15134: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15135: begin
15136:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15137:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15138:  Function wwStrToDate( const S : string) : boolean');
15139:  Function wwStrToTime( const S : string) : boolean');
15140:  Function wwStrToDateTime( const S : string) : boolean');
15141:  Function wwStrToTimeVal( const S : string) : TDateTime');
15142:  Function wwStrToDateVal( const S : string) : TDateTime');
15143:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15144:  Function wwStrToInt( const S : string) : boolean');
15145:  Function wwStrToFloat( const S : string) : boolean');
15146:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15147:  Function wwNextDay( Year, Month, Day : Word) : integer');
15148:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15149:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15150:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15151:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15152:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15153:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15154:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15155:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15156:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15157: end;
15158:
15159: unit uPSI_Themes;
15160: Function ThemeServices : TThemeServices');
15161: Function ThemeControl( AControl : TControl) : Boolean');
15162: Function UnthemedDesigner( AControl : TControl) : Boolean');
15163: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15164: begin
15165:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15166: end;
15167: Unit uPSC_menus;
15168:  Function StripHotkey( const Text : string) : string');
15169:  Function GetHotkey( const Text : string) : string');
15170:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15171:  Function IsAltGRPressed : boolean');
15172:
15173: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15174: begin
15175:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15176:   SIRegister_TIdIMAP4Server(CL);
15177: end;
15178:
15179: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15180: begin
15181:  'HASH_SIZE','LongInt'( 256);
15182:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
```

```
15183:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15184:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15185:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15186:    +' : Integer; Value : Variant; end');
15187:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15188:   SIRegister_TVariantSymbolTable(CL);
15189: end;
15190:
15191: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15192: begin
15193:   SIRegister_TThreadLocalVariables(CL);
15194:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15195:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15196:  Function ThreadLocals : TThreadLocalVariables');
15197:  Procedure WriteDebug( sz : String)');
15198:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15199:  'UDF_FAILURE','LongInt'( 1);
15200:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15201:  CL.AddTypeS('mTByteArray', 'array of byte;');
15202:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15203:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15204:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15205:  function IsNetworkConnected: Boolean;
15206:  function IsInternetConnected: Boolean;
15207:  function IsCOMConnected: Boolean;
15208:  function IsNetworkOn: Boolean;
15209:  function IsInternetOn: Boolean;
15210:  function IsCOMOn: Boolean;
15211:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15212:  TmrProc', 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);');
15213:  Function SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT');
15214:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15215:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15216:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15217:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15218:  Function GetMenu( hWnd : HWND) : HMENU');
15219:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15220: end;
15221:
15222: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15223: begin
15224:   SIRegister_IDataBlock(CL);
15225:   SIRegister_ISendDataBlock(CL);
15226:   SIRegister_ITransport(CL);
15227:   SIRegister_TDataBlock(CL);
15228:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15229:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15230:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15231:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15232:   SIRegister_TCustomDataBlockInterpreter(CL);
15233:   SIRegister_TSendDataBlock(CL);
15234:   'CallSig','LongWord')( $D800);
15235:   'ResultSig','LongWord')( $D400);
15236:   'asMask','LongWord')( $00FF);
15237:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15238:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15239:  Procedure CheckSignature( Sig : Integer)');
15240: end;
15241:
15242: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15243: begin
15244:   //CL.AddTypeS('HINTERNET', '___Pointer');
15245:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15246:   CL.AddTypeS('HINTERNET', 'Integer');
15247:   CL.AddTypeS('HINTERNET2', '___Pointer');
15248:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15249:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15250:   CL.AddTypeS('INTERNET_PORT', 'Word');
15251:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15252:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15253:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15254:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15255:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15256:  Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
       lpUrlComponents:TURLComponents):BOOL;
15257:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
       dwUrlLength:DWORD):BOOL;
15258:  Function InternetCloseHandle(hInet : HINTERNET) : BOOL');
15259:  Function
       InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
       lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15260:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15261:  ;dwContext:DWORD):HINTERNET;
15262:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
       lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15263:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
       dwContext:DWORD):BOOL;
15264:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15265:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
```

```
15266:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15267:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15268:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15269:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15270:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15271:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
        lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15272:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
        lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15273:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15274:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15275:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
        dwContext:DWORD):BOOL;
15276:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
        TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15277:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
        BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15278:  Function
        WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15279:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15280:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15281:  Function
        FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15282:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15283:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15284:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15285:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15286:  Function
        FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15287:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15288:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15289:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15290:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15291:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15292:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15293:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15294:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15295:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15296:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15297:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15298:  Function
        GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
        PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15299:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15300:  Function
        GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15301:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
        PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15302:  Function
        HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15303:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
        dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15304:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15305:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15306:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15307:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15308:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15309:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15310:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15311:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
        lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15312:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
        lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15313:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15314:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15315:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15316:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15317: end;
15318:
15319: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15320: begin
15321:   AddTypeS('strCharSet', 'set of char');
15322:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15323:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15324:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15325:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15326:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15327:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15328:  Procedure strStripWhiteSpace( var s : string)');
15329:  Function strRemoveChar( str : string; removeChar : char) : string');
15330:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15331:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15332:  Function wwEqualStr( s1, s2 : string) : boolean');
15333:  Function strCount( s : string; delimeter : char) : integer');
15334:  Function strWhiteSpace : strCharSet');
15335:  Function wwExtractFileNameOnly( const FileName : string) : string');
15336:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15337:  Function strTrailing( s : string; delimeter : char) : string');
```

```
15338:  Function strPreceding( s : string; delimeter : char) : string');
15339:  Function wwstrReplace( s, Find, Replace : string) : string');
15340: end;
15341:
15342: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15343: begin
15344:   SIRegister_TRemoteDataModule(CL);
15345:   SIRegister_TCRemoteDataModule(CL);
15346:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15347:  Procedure UnregisterPooled( const ClassID : string)');
15348:  Procedure EnableSocketTransport( const ClassID : string)');
15349:  Procedure DisableSocketTransport( const ClassID : string)');
15350:  Procedure EnableWebTransport( const ClassID : string)');
15351:  Procedure DisableWebTransport( const ClassID : string)');
15352: end;
15353:
15354: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15355: begin
15356:  Function mxArcCos( x : Real) : Real');
15357:  Function mxArcSin( x : Real) : Real');
15358:  Function Comp2Str( N : Comp) : String');
15359:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15360:  Function Int2Str( N : LongInt) : String');
15361:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15362:  Function LogXY( x, y : Real) : Real');
15363:  Function Pennies2Dollars( C : Comp) : String');
15364:  Function mxPower( X : Integer; Y : Integer) : Real');
15365:  Function Real2Str( N : Real; Width, Places : integer) : String');
15366:  Function mxStr2Comp( MyString : string) : Comp');
15367:  Function mxStr2Pennies( S : String) : Comp');
15368:  Function Str2Real( MyString : string) : Real');
15369:  Function XToTheY( x, y : Real) : Real');
15370: end;
15371:
15372: //************************Cindy Functions!*****************************
15373: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15374: begin
15375:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15376:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15377:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15378:  MessagePlainText','String 'text/plain');
15379:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15380:  MessageAlterText_Html','String 'multipart/alternative');
15381:  MessageHtml_Attach','String 'multipart/mixed');
15382:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15383:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15384:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15385:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15386:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15387:  Function ForceDecodeHeader( aHeader : String) : String');
15388:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15389:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15390:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15391:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15392:  Function Get_MD5( const aFileName : string) : string');
15393:  Function Get_MD5FromString( const aString : string) : string');
15394: end;
15395:
15396: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15397: begin
15398:  Function IsFolder( SRec : TSearchrec) : Boolean');
15399:  Function isFolderReadOnly( Directory : String) : Boolean');
15400:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15401:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15402:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15403:  Function DiskFreeBytes( Drv : Char) : Int64');
15404:  Function DiskBytes( Drv : Char) : Int64');
15405:  Function GetFileBytes( Filename : String) : Int64');
15406:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15407:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15408:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15409:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15410:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15411:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15412:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15413:  SE_TCB_NAME','String 'SeTcbPrivilege');
15414:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15415:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15416:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15417:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15418:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15419:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15420:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15421:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15422:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15423:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15424:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15425:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15426:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
```

```
15427:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15428:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15429:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15430:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15431:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15432:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15433:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15434:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15435: end;
15436:
15437:
15438: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15439: begin
15440:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15441:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15442:  Function ShellGetExtensionName( FileName : String) : String');
15443:  Function ShellGetIconIndex( FileName : String) : Integer');
15444:  Function ShellGetIconHandle( FileName : String) : HIcon');
15445:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15446:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15447:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15448:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15449:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15450:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15451:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15452:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
       WaitCloseCompletion : boolean) : Boolean');
15453:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15454:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15455:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15456:  Function GetModificationDate( Filename : String) : TDateTime');
15457:  Function GetCreationDate( Filename : String) : TDateTime');
15458:  Function GetLastAccessDate( Filename : String) : TDateTime');
15459:  Function FileDelete( Filename : String) : Boolean');
15460:  Function FileIsOpen( Filename : string) : boolean');
15461:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15462:  Function DirectoryDelete( Directory : String) : Boolean');
15463:  Function GetPrinters( PrintersList : TStrings) : Integer');
15464:  Procedure SetDefaultPrinter( PrinterName : String)');
15465:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15466:  Function WinToDosPath( WinPathName : String) : String');
15467:  Function DosToWinPath( DosPathName : String) : String');
15468:  Function cyGetWindowsVersion : TWindowsVersion');
15469:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15470:  Procedure WindowsShutDown( Restart : boolean)');
15471:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
       FileIcon:string;NumIcone:integer);
15472:  Procedure GetWindowsFonts( FontsList : TStrings)');
15473:  Function GetAvailableFilename( DesiredFileName : String): String');
15474: end;
15475:
15476: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15477: begin
15478:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15479:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15480:   Type(TStringRead', '( srFromLeft, srFromRight )');
15481:   Type(TStringReads', 'set of TStringRead');
15482:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15483:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15484:   Type(TWordsOptions', 'set of TWordsOption');
15485:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15486:   Type(TCarTypes', 'set of TCarType');
15487:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15488:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15489:  Function Char_GetType( aChar : Char) : TCarType');
15490:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15491:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15492:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15493:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15494:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15495:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15496:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15497:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15498:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15499:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15500:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15501:  Function String_Quote( Str : String) : String');
15502:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15503:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15504:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15505:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15506:  Function String_ToInt( Str : String) : Integer');
15507:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15508:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15509:  Function String_Reverse( Str : String) : String');
15510:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15511:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15512:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
```

```
15513:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15514:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
        Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15515:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15516:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15517:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15518:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15519:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15520:  Function String_End( Str : String; Cars : Word) : String');
15521:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
        AlwaysFindFromBeginning : Boolean) : String');
15522:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15523:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15524:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15525:  Function String_IsNumbers( Str : String) : Boolean');
15526:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15527:  Function StringToCsvCell( aStr : String) : String');
15528: end;
15529:
15530: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15531: begin
15532:  Function LongDayName( aDate : TDate) : String');
15533:  Function LongMonthName( aDate : TDate) : String');
15534:  Function ShortYearOf( aDate : TDate) : byte');
15535:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15536:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15537:  Function SecondsToMinutes( Seconds : Integer) : Double');
15538:  Function MinutesToSeconds( Minutes : Double) : Integer');
15539:  Function MinutesToHours( Minutes : Integer) : Double');
15540:  Function HoursToMinutes( Hours : Double) : Integer');
15541:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15542:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15543:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15544:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15545:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15546:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15547:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
        RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15548:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15549:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15550: end;
15551:
15552: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15553: begin
15554:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15555:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15556:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15557:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15558:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15559:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15560:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15561:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15562:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15563:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15564:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15565:  Function
        TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15566:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15567:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
        CopySubChildren:Bool;
15568:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15569:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15570:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15571:  Procedure cyCenterControl( aControl : TControl)');
15572:  Function GetLastParent( aControl : TControl) : TWinControl');
15573:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15574:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15575: end;
15576:
15577: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15578: begin
15579:  Function TablePackTable( Tab : TTable) : Boolean');
15580:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15581:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15582:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15583:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15584:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15585:  Function TableEmptyTable( Tab : TTable) : Boolean');
15586:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15587:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15588:  Function
        TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Bool):TTable;
15589:  Function
        TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15590:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15591: end;
15592:
15593: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15594: begin
```

```
15595:   SIRegister_TcyRunTimeDesign(CL);
15596:   SIRegister_TcyShadowText(CL);
15597:   SIRegister_TcyBgPicture(CL);
15598:   SIRegister_TcyGradient(CL);
15599:   SIRegister_tcyBevel(CL);
15600:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15601:   SIRegister_tcyBevels(CL);
15602:   SIRegister_TcyImagelistOptions(CL);
15603:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15604: end;
15605:
15606: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15607: begin
15608:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
       adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
       Maxdegrade : Byte;'+
15609:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15610:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15611:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15612:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15613:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15614:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15615:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15616:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
       BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15617:  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;
15618:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15619:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15620:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
       TColor; aState : TButtonState; Focused, Hot : Boolean)');
15621:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
       GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15622:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
       const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15623:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
       TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15624:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
       TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15625:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
       WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15626:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15627:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15628:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15629:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
       CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15630:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15631:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15632:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15633:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15634:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15635:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15636:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15637:  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);
15638:  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;
15639:  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);');
15640:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15641:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15642:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15643:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15644:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15645:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15646:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15647:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15648:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15649:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15650:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15651:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15652:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15653:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15654:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15655:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15656: end;
15657:
```

```
15658: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15659: begin
15660:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15661:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15662:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15663:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15664:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15665:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15666:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15667:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15668:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15669:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15670:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15671:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
      bmInvertReverseFromColor);
15672:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
      rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15673:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15674:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15675: end;
15676:
15677: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15678: begin
15679:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15680:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15681:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15682:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15683:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15684:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15685:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15686:  Const SC_GROUP_IDENTIFIERW','String') '+');
15687:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15688:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15689:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15690:  Const SERVICE_INACTIVE','LongWord $00000002);
15691:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15692:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15693:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15694:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15695:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15696:  Const SERVICE_STOPPED','LongWord $00000001);
15697:  Const SERVICE_START_PENDING','LongWord $00000002);
15698:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15699:  Const SERVICE_RUNNING','LongWord $00000004);
15700:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15701:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15702:  Const SERVICE_PAUSED','LongWord $00000007);
15703:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15704:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15705:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15706:  Const SC_MANAGER_CONNECT','LongWord $0001);
15707:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15708:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15709:  Const SC_MANAGER_LOCK','LongWord $0008);
15710:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15711:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15712:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15713:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15714:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15715:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15716:  Const SERVICE_START','LongWord $0010);
15717:  Const SERVICE_STOP','LongWord $0020);
15718:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15719:  Const SERVICE_INTERROGATE','LongWord $0080);
15720:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15721:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15722:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15723:  Const SERVICE_ADAPTER','LongWord $00000004);
15724:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15725:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15726:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15727:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15728:  Const SERVICE_BOOT_START','LongWord $00000000);
15729:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15730:  Const SERVICE_AUTO_START','LongWord $00000002);
15731:  Const SERVICE_DEMAND_START','LongWord $00000003);
15732:  Const SERVICE_DISABLED','LongWord $00000004);
15733:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15734:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15735:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15736:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15737:   CL.AddTypeS('SC_HANDLE', 'THandle');
15738:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15739:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15740:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15741:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15742:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15743:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15744:   Const TServiceStatus', '_SERVICE_STATUS');
```

```
15745:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15746:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15747:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15748:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15749:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15750:   TEnumServiceStatus', 'TEnumServiceStatusA');
15751:   SC_LOCK', '___Pointer');
15752:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15753:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15754:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15755:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15756:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15757:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15758:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15759:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15760:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15761:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15762:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15763:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15764:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15765:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15766:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15767:   TQueryServiceConfig', 'TQueryServiceConfigA');
15768:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15769:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15770:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15771:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15772:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15773:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15774:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15775:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
      TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15776:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
      lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
      : DWORD):BOOL');
15777:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
      lpcchBuffer:DWORD):BOOL');
15778:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
      lpcchBuffer:DWORD):BOOL;
15779:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15780:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15781:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15782:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15783:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
      cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15784:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15785:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15786:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15787:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15788: end;
15789:
15790: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15791: begin
15792:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
      AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
      BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15793:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
      DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15794:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15795:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
      TWinControl;
15796:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
      AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15797:  Function CreateNotifyThread(const
      FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15798: end;
15799:
15800: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15801: begin
15802:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15803:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15804:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15805:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15806:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15807:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15808:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15809:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15810:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15811:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15812:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15813:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15814:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15815:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15816:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15817:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15818:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15819:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
```

```
15820:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15821:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15822:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15823:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15824:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15825:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15826:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15827:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15828:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15829:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15830:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15831:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15832:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15833:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15834:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15835:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15836:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15837:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15838:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15839:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15840:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15841:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15842:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15843:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15844:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15845:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15846:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15847:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
      List:TStrings):Bool
15848:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15849:   FindClass('TOBJECT'),'EJclFileSummaryError');
15850:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15851:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15852:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15853:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15854:   SIRegister_TJclFilePropertySet(CL);
15855:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15856:   SIRegister_TJclFileSummary(CL);
15857:   SIRegister_TJclFileSummaryInformation(CL);
15858:   SIRegister_TJclDocSummaryInformation(CL);
15859:   SIRegister_TJclMediaFileSummaryInformation(CL);
15860:   SIRegister_TJclMSISummaryInformation(CL);
15861:   SIRegister_TJclShellSummaryInformation(CL);
15862:   SIRegister_TJclStorageSummaryInformation(CL);
15863:   SIRegister_TJclImageSummaryInformation(CL);
15864:   SIRegister_TJclDisplacedSummaryInformation(CL);
15865:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15866:   SIRegister_TJclMiscSummaryInformation(CL);
15867:   SIRegister_TJclWebViewSummaryInformation(CL);
15868:   SIRegister_TJclMusicSummaryInformation(CL);
15869:   SIRegister_TJclDRMSummaryInformation(CL);
15870:   SIRegister_TJclVideoSummaryInformation(CL);
15871:   SIRegister_TJclAudioSummaryInformation(CL);
15872:   SIRegister_TJclControlPanelSummaryInformation(CL);
15873:   SIRegister_TJclVolumeSummaryInformation(CL);
15874:   SIRegister_TJclShareSummaryInformation(CL);
15875:   SIRegister_TJclLinkSummaryInformation(CL);
15876:   SIRegister_TJclQuerySummaryInformation(CL);
15877:   SIRegister_TJclImageInformation(CL);
15878:   SIRegister_TJclJpegSummaryInformation(CL);
15879: end;
15880:
15881: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15882: begin
15883:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15884:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15885:   T8087Infinity','( icProjective, icAffine )');
15886:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15887:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15888:  Function Get8087ControlWord : Word');
15889:  Function Get8087Infinity : T8087Infinity');
15890:  Function Get8087Precision : T8087Precision');
15891:  Function Get8087Rounding : T8087Rounding');
15892:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15893:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15894:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15895:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15896:  Function Set8087ControlWord( const Control : Word) : Word');
15897:  Function ClearPending8087Exceptions : T8087Exceptions');
15898:  Function GetPending8087Exceptions : T8087Exceptions');
15899:  Function GetMasked8087Exceptions : T8087Exceptions');
15900:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15901:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15902:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15903: end;
15904:
15905: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15906: begin
15907:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
```

```
15908:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15909:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
15910:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15911:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15912:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15913:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15914:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15915: end;
15916:
15917: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15918: begin
15919:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15920:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15921:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15922:   type ULONG', 'Cardinal');
15923:      LPCWSTR', 'PChar');
15924:   CL.AddTypeS('LPWSTR', 'PChar');
15925:   LPSTR', 'PChar');
15926:   TBindVerb', 'ULONG');
15927:   TBindInfoF', 'ULONG');
15928:   TBindF', 'ULONG');
15929:   TBSCF', 'ULONG');
15930:   TBindStatus', 'ULONG');
15931:   TCIPStatus', 'ULONG');
15932:   TBindString', 'ULONG');
15933:   TPiFlags', 'ULONG');
15934:   TOIBdgFlags', 'ULONG');
15935:   TParseAction', 'ULONG');
15936:   TPSUAction', 'ULONG');
15937:   TQueryOption', 'ULONG');
15938:   TPUAF', 'ULONG');
15939:   TSZMFlags', 'ULONG');
15940:   TUrlZone', 'ULONG');
15941:   TUrlTemplate', 'ULONG');
15942:   TZAFlags', 'ULONG');
15943:   TUrlZoneReg', 'ULONG');
15944:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15945:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15946:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15947:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15948:  const 'CF_NULL','LongInt').SetInt( 0);
15949:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15950:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15951:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15952:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15953:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15954:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15955:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15956:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15957:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15958:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15959:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15960:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15961:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15962:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15963:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15964:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15965:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15966:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15967:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15968:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15969:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15970:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15971:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15972:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15973:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15974:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15975:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15976:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15977:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15978:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15979:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15980:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15981:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15982:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15983:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15984:   SIRegister_IPersistMoniker(CL);
15985:   SIRegister_IBindProtocol(CL);
15986:   SIRegister_IBinding(CL);
15987:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15988:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15989:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15990:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15991:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15992:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15993:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15994:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15995:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
```

```
15996:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15997:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15998:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15999:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
16000:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
16001:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
16002:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
16003:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
16004:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
16005:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
16006:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
16007:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
16008:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
16009:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
16010:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
16011:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
16012:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
16013:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
16014:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
16015:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
16016:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
16017:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
16018:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
16019:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
16020:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16021:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16022:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16023:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16024:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16025:  const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16026:  const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
16027:  const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
16028:  const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16029:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16030:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16031:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16032:  const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
16033:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
16034:  const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16035:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16036:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16037:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16038:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16039:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16040:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16041:  const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16042:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16043:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16044:  const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
16045:  const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
16046:  const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
16047:  const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16048:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16049:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16050:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16051:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16052:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16053:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16054:  const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
16055:  const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16056:  const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
16057:  const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16058:  const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16059:  const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16060:  const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
16061:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16062:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16063:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16064:  const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
16065:  const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
16066:  const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16067:  const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16068:  const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16069:  const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16070:  const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16071:  // PBindInfo', '^TBindInfo // will not work');
16072:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16073:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16074:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16075:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16076:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16077:   TBindInfo', '_tagBINDINFO');
16078:   BINDINFO', '_tagBINDINFO');}
16079:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16080:    +'criptor : DWORD; bInheritHandle : BOOL; end');
16081:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16082:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16083:   //PRemBindInfo', '^TRemBindInfo // will not work');
16084:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
```

```
16085:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16086:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16087:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16088:    +'n; dwReserved : DWORD; end');
16089:   TRemBindInfo', '_tagRemBINDINFO');
16090:   RemBINDINFO', '_tagRemBINDINFO');}
16091:   //PRemFormatEtc', '^TRemFormatEtc // will not work');
16092:   tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16093:   TRemFormatEtc', 'tagRemFORMATETC');
16094:   RemFORMATETC', 'tagRemFORMATETC');
16095:   SIRegister_IBindStatusCallback(CL);
16096:   SIRegister_IAuthenticate(CL);
16097:   SIRegister_IHttpNegotiate(CL);
16098:   SIRegister_IWindowForBindingUI(CL);
16099:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16100:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16101:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16102:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16103:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16104:  const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16105:  const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT',LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16106:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16107:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16108:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16109:   SIRegister_ICodeInstall(CL);
16110:   SIRegister_IWinInetInfo(CL);
16111:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16112:   SIRegister_IHttpSecurity(CL);
16113:   SIRegister_IWinInetHttpInfo(CL);
16114:   SIRegister_IBindHost(CL);
16115:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16116:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16117:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16118:  Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback) : HResult');
16119:  Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback) : HResult');
16120:  Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
       : IBindStatusCallback) : HResult');
16121:  Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
      IBindStatusCallback) : HResult');
16122:  Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
      p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16123:  Function HlinkGoBack( unk : IUnknown) : HResult');
16124:  Function HlinkGoForward( unk : IUnknown) : HResult');
16125:  Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult');
16126: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult');
16127:   SIRegister_IInternet(CL);
16128:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16129:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16130:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16131:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16132:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16133:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16134:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16135:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16136:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16137:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16138:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16139:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16140:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16141:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16142:   //POLEStrArray', '^TOLESTRArray // will not work');
16143:   SIRegister_IInternetBindInfo(CL);
16144:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16145:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16146:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16147:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16148:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16149:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16150:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16151:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16152:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16153:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16154:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16155:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16156:   //PProtocolData', '^TProtocolData // will not work');
16157:   _tagPROTOCOLDATA','record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16158:   TProtocolData', '_tagPROTOCOLDATA');
16159:   PROTOCOLDATA', '_tagPROTOCOLDATA');
16160:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink, 'IInternetProtocolSink');
16161:   SIRegister_IInternetProtocolRoot(CL);
16162:   SIRegister_IInternetProtocol(CL);
16163:   SIRegister_IInternetProtocolSink(CL);
16164:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16165:   SIRegister_IInternetSession(CL);
16166:   SIRegister_IInternetThreadSwitch(CL);
16167:   SIRegister_IInternetPriority(CL);
16168:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16169:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16170:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
```

```
16171:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16172:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16173:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16174:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16175:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16176:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16177:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16178:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16179:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16180:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16181:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16182:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16183:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16184:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16185:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16186:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16187:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16188:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16189:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16190:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16191:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16192:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16193:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16194:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16195:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16196:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16197:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16198:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16199:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16200:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16201:   SIRegister_IInternetProtocolInfo(CL);
16202:   IOInet', 'IInternet');
16203:   IOInetBindInfo', 'IInternetBindInfo');
16204:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16205:   IOInetProtocol', 'IInternetProtocol');
16206:   IOInetProtocolSink', 'IInternetProtocolSink');
16207:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16208:   IOInetSession', 'IInternetSession');
16209:   IOInetPriority', 'IInternetPriority');
16210:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16211:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16212:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16213:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16214:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16215:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16216:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16217:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16218:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16219:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16220:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16221:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16222:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16223:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16224:    //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16225:   //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16226: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16227: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16228:   //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16229:   //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16230:   //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16231:   const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16232:    SIRegister_IInternetSecurityMgrSite(CL);
16233:   const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16234:   const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16235:   const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16236:   const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16237:   const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16238:   const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16239:   const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16240:   const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16241:   const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16242:   const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16243:   const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16244:   const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16245:   const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16246:   const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16247:   const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16248:   const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16249:   const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16250:   const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
```

```
16251:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16252:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16253:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16254:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16255:  const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16256:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16257:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16258:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16259:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16260:   SIRegister_IInternetSecurityManager(CL);
16261:   SIRegister_IInternetHostSecurityManager(CL);
16262:   SIRegister_IInternetSecurityManagerEx(CL);
16263:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16264:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16265:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16266:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16267:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16268:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16269:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16270:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16271:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16272:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16273:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16274:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16275:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16276:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16277:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16278:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16279:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16280:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16281:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16282:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16283:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16284:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16285:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16286:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16287:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16288:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16289:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16290:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16291:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16292:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16293:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16294:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16295:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16296:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16297:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16298:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16299:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16300:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16301:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16302:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16303:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16304:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16305:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16306:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16307:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16308:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16309:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16310:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16311:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16312:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16313:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16314:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16315:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16316:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16317:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16318:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16319:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16320:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16321:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16322:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16323:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16324:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16325:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16326:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16327:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16328:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16329:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16330:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16331:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16332:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16333:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16334:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16335:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16336:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16337:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16338:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16339:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
```

```
16340:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16341:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16342:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16343:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16344:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16345:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16346:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16347:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16348:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16349:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16350:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16351:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16352:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16353:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16354:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16355:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16356:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16357:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16358:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16359:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16360:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16361:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16362:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16363:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16364:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16365:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16366:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16367:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16368:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16369:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16370:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16371:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16372:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16373:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16374:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16375:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16376:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16377:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16378:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16379:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16380:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16381:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16382:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16383:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16384:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16385:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16386:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16387:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16388:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16389:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16390:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16391:   //PZoneAttributes', '^TZoneAttributes // will not work');
16392:  _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');
16393:   { _ZONEATTRIBUTES = packed record
16394:     cbSize: ULONG;
16395:     szDisplayName: array [0..260 - 1] of WideChar;
16396:     szDescription: array [0..200 - 1] of WideChar;
16397:     szIconPath: array [0..260 - 1] of WideChar;
16398:     dwTemplateMinLevel: DWORD;
16399:     dwTemplateRecommended: DWORD;
16400:     dwTemplateCurrentLevel: DWORD;
16401:     dwFlags: DWORD;
16402:   end; }
16403:   TZoneAttributes', '_ZONEATTRIBUTES');
16404:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16405:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16406:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16407:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16408:   SIRegister_IInternetZoneManager(CL);
16409:   SIRegister_IInternetZoneManagerEx(CL);
16410:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16411:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16412:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16413:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16414:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16415:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16416:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16417:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16418:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16419:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16420:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16421:   TCodeBaseHold', '_tagCODEBASEHOLD');
16422:   CODEBASEHOLD', '_tagCODEBASEHOLD');
16423:   //PSoftDistInfo', '^TSoftDistInfo // will not work');
16424:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16425:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16426:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
```

```
16427:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16428:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16429:   TSoftDistInfo', '_tagSOFTDISTINFO');
16430:   SOFTDISTINFO', '_tagSOFTDISTINFO');
16431:   SIRegister_ISoftDistExt(CL);
16432:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16433:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
       dwAdvertisedVersionLS : DWORD) : HResult');
16434:   SIRegister_IDataFilter(CL);
16435:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16436:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
16437:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16438:    +'terFlags : DWORD; end');
16439:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16440:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16441:   //PDataInfo', '^TDataInfo // will not work');
16442:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16443:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16444:   TDataInfo', '_tagDATAINFO');
16445:   DATAINFO', '_tagDATAINFO');
16446:   SIRegister_IEncodingFilterFactory(CL);
16447:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16448:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16449:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16450:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16451:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16452:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16453:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16454:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16455:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16456: end;
16457:
16458: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16459: begin
16460:  Procedure reformatMemo( const m : TCustomMemo)');
16461:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16462:  Procedure MoveToTop( memo : TMemo)');
16463:  Procedure ScrollToTop( memo : TMemo)');
16464:  Function LineNumberClicked( memo : TMemo) : integer');
16465:  Function MemoClickedLine( memo : TMemo) : integer');
16466:  Function ClickedMemoLine( memo : TMemo) : integer');
16467:  Function MemoLineClicked( memo : TMemo) : integer');
16468:  Function LinePositionClicked( Memo : TMemo) : integer');
16469:  Function ClickedMemoPosition( memo : TMemo) : integer');
16470:  Function MemoPositionClicked( memo : TMemo) : integer');
16471:  Procedure AdjustGridSize( grid : TDrawGrid)');
16472:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16473:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16474:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16475:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16476:  Procedure sortstrDown( var s : string)');
16477:  Procedure sortstrUp( var s : string)');
16478:  Procedure rotatestrleft( var s : string)');
16479:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16480:  Function deblank( s : string) : string');
16481:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16482:  Procedure FreeAndClearListBox( C : TListBox);');
16483:  Procedure FreeAndClearMemo( C : TMemo);');
16484:  Procedure FreeAndClearStringList( C : TStringList);');
16485:  Function dffgetfilesize( f : TSearchrec) : int64');
16486: end;
16487:
16488: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16489: begin
16490:   CL.AddTypeS('intset', 'set of byte');
16491:   TPoint64', 'record x : int64; y : int64; end');
16492:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16493:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16494:  Function GeneratePentagon( n : integer) : integer');
16495:  Function IsPentagon( p : integer) : boolean');
16496:  Function isSquare( const N : int64) : boolean');
16497:  Function isCube( const N : int64) : boolean');
16498:  Function isPalindrome( const n : int64) : boolean;');
16499:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16500:  Function GetEulerPhi( n : int64) : int64');
16501:  Function dffIntPower( a, b : int64) : int64;');
16502:  Function IntPower1( a : extended; b : int64) : extended;');
16503:  Function gcd2( a, b : int64) : int64');
16504:  Function GCDMany( A : array of integer) : integer');
16505:  Function LCMMany( A : array of integer) : integer');
16506:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16507:  Function dffFactorial( n : int64) : int64');
16508:  Function digitcount( n : int64) : integer');
16509:  Function nextpermute( var a : array of integer) : boolean');
16510:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16511:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16512:  Function InttoBinaryStr( nn : integer) : string');
16513:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16514:  Function AngleToStr( angle : extended) : string');
```

```
16515:  Function deg2rad( deg : extended) : extended');
16516:  Function rad2deg( rad : extended) : extended');
16517:  Function GetLongToMercProjection( const long : extended) : extended');
16518:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16519:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16520:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16521:   SIRegister_TPrimes(CL);
16522:   //RIRegister_TPrimes(CL);
16523:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16524:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16525:  Function Random64( const N : Int64) : Int64;');
16526:  Procedure Randomize64');
16527:  Function Random641 : extended;');
16528:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16529: end;
16530:
16531: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16532: begin
16533:   TrealPoint', 'record x : extended; y : extended; end');
16534:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16535:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16536:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16537:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16538:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16539:  Function realpoint( x, y : extended) : TRealPoint');
16540:  Function dist( const p1, p2 : TrealPoint) : extended');
16541:  Function intdist( const p1, p2 : TPoint) : integer');
16542:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16543:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16544:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16545:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16546:  Function GetTheta( const L : TLine) : extended;');
16547:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16548:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16549:  Procedure Extendline( var L : TLine; dist : integer);');
16550:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16551:  Function Linesintersect( line1, line2 : TLine) : boolean');
16552:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16553:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16554:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16555:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16556:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16557:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16558:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16559:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16560:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16561:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16562:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16563:  Function DegtoRad( d : extended) : extended');
16564:  Function RadtoDeg( r : extended) : extended');
16565:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16566:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16567:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16568:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16569:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16570:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16571:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16572:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16573:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16574: end;
16575:
16576:
16577: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16578: begin
16579:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16580:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16581:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16582:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16583:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16584:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16585:    +'arth : extended; Phase : extended; end');
16586:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16587:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16588:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16589:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16590:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16591:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16592:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16593:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16594:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16595:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
       ApparentRaDecl:TRPoint; end');
16596:   SIRegister_TAstronomy(CL);
```

```
16597:  Function AngleToStr( angle : extended) : string');
16598:  Function StrToAngle( s : string; var angle : extended) : boolean');
16599:  Function HoursToStr24( t : extended) : string');
16600:  Function RPoint( x, y : extended) : TRPoint');
16601:  Function getStimename( t : TDTType) : string');
16602: end;
16603:
16604: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16605: begin
16606:   TCardValue', 'Integer');
16607:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16608:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16609:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16610:   SIRegister_TCard(CL);
16611:   SIRegister_TDeck(CL);
16612: end;
16613:
16614: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16615: begin
16616:   tMethodCall', 'Procedure');
16617:   tVerboseCall', 'Procedure ( s : string)');
16618:  // PTEdge', '^TEdge // will not work');
16619:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16620:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16621:   SIRegister_TNode(CL);
16622:   SIRegister_TGraphList(CL);
16623: end;
16624:
16625: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16626: begin
16627:   ParserFloat', 'extended');
16628:   //PParserFloat', '^ParserFloat // will not work');
16629:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16630:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16631:   //POperation', '^TOperation // will not work');
16632:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16633:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16634:    +'; Token : TDFFToken; end');
16635:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16636:   (CL.FindClass('TOBJECT'),'EMathParserError');
16637:   CL.FindClass('TOBJECT'),'ESyntaxError');
16638:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16639:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16640:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16641:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16642:   (CL.FindClass('TOBJECT'),'EBadName');
16643:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16644:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16645:   SIRegister_TCustomParser(CL);
16646:   SIRegister_TExParser(CL);
16647: end;
16648:
16649:   function isService: boolean;
16650:   begin
16651:     result:= NOT(Application is TApplication);
16652:     {result:= Application is TServiceApplication;}
16653:   end;
16654:   function isApplication: boolean;
16655:   begin
16656:     result:= Application is TApplication;
16657:   end;
16658:   //SM_REMOTESESSION = $1000
16659:   function isTerminalSession: boolean;
16660:   begin
16661:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16662:   end;
16663:
16664: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16665: begin
16666:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16667:    +'String; margin_bottom : String; margin_left : String; margin_right : Strin'
16668:    +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16669:  Function cyURLEncode( const S : string) : string');
16670:  Function MakeResourceURL(const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16671:  Function MakeResourceURL1(const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;;
16672:  Function cyColorToHtml( aColor : TColor) : String');
16673:  Function HtmlToColor( aHtmlColor : String) : TColor');
16674:  //Function GetStreamEncoding( aStream : TStream) : TEncoding');
16675:  // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean');
16676:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
16677:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16678:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16679:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16680:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16681:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16682:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16683:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16684:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16685:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
```

```
16686: end;
16687:
16688:
16689: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16690: begin
16691:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16692:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16693:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16694:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16695:    +'inationsRepeat, CombinationsRepeatDown )');
16696:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16697:   SIRegister_TComboSet(CL);
16698: end;
16699:
16700:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16701: begin
16702:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16703:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16704:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16705:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16706:    +'mmHandle : THandle; aString : String; userParam : Integer)');
16707:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16708:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16709:   SIRegister_TcyBaseComm(CL);
16710:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16711:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16712:  Function ValidateFileMappingName( aName : String) : String');
16713:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16714: end;
16715:
16716: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16717: begin
16718:   CL.AddTypeS('DERString', 'String');
16719:   CL.AddTypeS('DERChar', 'Char');
16720:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16721:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16722:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16723:   CL.AddTypeS('DERNString', 'String');
16724:  const DERDecimalSeparator','String').SetString( '.');
16725:  const DERDefaultChars','String')('+º@/%-
      _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16726:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16727:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16728:  Function isValidWebMailChar( aChar : Char) : Boolean');
16729:  Function isValidwebSite( aStr : String) : Boolean');
16730:  Function isValidWebMail( aStr : String) : Boolean');
16731:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16732:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16733:  Function IsDERChar( aChar : Char) : Boolean');
16734:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16735:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16736:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16737:  Function IsDERSymbols( aDERString : String) : Boolean');
16738:  Function StringToDERCharSet( aStr : String) : DERString');
16739:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16740:  Function IsDERNChar( aChar : Char) : Boolean');
16741:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16742:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
16743:  Function DERExtractWebMail( aDERStr : DERString) : String');
16744:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16745:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16746:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
      RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16747:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
      aElementsType : TElementsType) : String;');
16748:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
      Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16749: end;
16750:
16751: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16752: begin
16753:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16754:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16755:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16756:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16757:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16758:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16759:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16760:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16761:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
      RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16762:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
      PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
      PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16763:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
      SingleDestinationColor, RefreshBmp : Boolean)');
16764:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
```

```
16765:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16766:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16767:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16768:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16769:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16770:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16771:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16772: end;
16773:
16774: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16775: begin
16776:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16777:   TPCMChannel', '( cMono, cStereo )');
16778:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16779:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16780:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
16781:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16782:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16783:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16784:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16785:    +'it48000Hz, Stereo16bit48000Hz )');
16786:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16787:    +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16788:    tWaveFormatEx', 'PWaveFormatEx');
16789:    HMMIO', 'Integer');
16790:   TWaveDeviceFormats', 'set of TPCMFormat');
16791:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16792:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16793:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16794:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16795:   TWaveOutOptions', 'set of TWaveOutOption');
16796:   TStreamOwnership2', '( soReference, soOwned )');
16797:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16798:  // PRawWave', '^TRawWave // will not work');
16799:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16800:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioError');
16801:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError');
16802:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation');
16803:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16804:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16805:    +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16806:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16807:    +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16808:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16809:    +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16810:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16811:    +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16812:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16813:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16814:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16815:  CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16816:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16817:  GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16818:  CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16819:  OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16820:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16821:  GetAudioFormat( FormatTag : Word) : String');
16822:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
16823:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16824:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16825:  GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16826:  InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16827:  SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16828:  ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16829:  MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
       TObject; BufferSize : DWORD) : Boolean');
16830:  ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
       dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16831:  SetPCMAudioFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
       TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16832:  Procedure SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16833:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16834:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');
16835:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) : String');
16836:  WaitForSyncObject( SyncObject : THandle; Timeout : DWORD) : DWORD');
16837:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT');
16838: end;
16839:
16840: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16841: begin
16842:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16843:  CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16844:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16845:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
16846:  'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16847:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16848:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16849:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16850:   CL.AddTypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg )');
```

```
16851:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16852:   SIRegister_TNamedPipe(CL);
16853:   SIRegister_TServerPipe(CL);
16854:   SIRegister_TClientPipe(CL);
16855:  CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD) : DWORD');
16856:  Function HasOverlappedIoCompleted( const ov : OVERLAPPED) : Boolean');
16857:  Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean):
       TOverlappedResult;
16858:  Function GetStreamAsText( stm : TStream) : string');
16859:  Procedure SetStreamAsText( const aTxt : string; stm : TStream)');
16860: end;
16861:
16862: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16863: begin
16864:  // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16865:   SIRegister_TThumbData(CL);
16866:  'PIC_BMP','LongInt').SetInt( 0);
16867:  'PIC_JPG','LongInt').SetInt( 1);
16868:  'THUMB_WIDTH','LongInt').SetInt( 60);
16869:  'THUMB_HEIGHT','LongInt').SetInt( 60);
16870:  Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean');
16871:  Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)');
16872:  Function ReadBitmap( Filehandle : Integer; Width, Height : Integer) : TBitmap');
16873:  Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap)');
16874:  Function OpenPicture( fn : string; var tp : Integer) : Integer');
16875:  Function ConvertPicture( pi : Integer; tp : Integer) : TBitmap');
16876:  Function LoadPicture( fn : string; var w, h : Integer) : TBitmap');
16877:  Function TurnBitmap( bmp : TBitmap; ang : Integer) : TBitmap');
16878:  Function RotateBitmap( Bitmap : TBitmap; Direction : Integer) : TBitmap');
16879:  Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect');
16880:  Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer) : TBitmap');
16881:  Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer)');
16882:  Procedure FindFiles( path, mask : string; items : TStringList)');
16883:  Function LetFileName( s : string) : string');
16884:  Function LetParentPath( path : string) : string');
16885:  Function AddBackSlash( path : string) : string');
16886:  Function CutBackSlash( path : string) : string');
16887: end;
16888:
16889: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16890: begin
16891:  //'BYTES','LongInt').SetInt( 1);
16892:  'KBYTES','LongInt').SetInt( 1024);
16893:  'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ));
16894:  'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ));
16895:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16896:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D}');
16897:   SIRegister_MakeComServerMethodsPublic(CL);
16898:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16899:  Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) : Boolean');
16900:  Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean)');
16901:  Function TBGetTempFolder : string');
16902:  Function TBGetTempFile : string');
16903:  Function TBGetModuleFilename : string');
16904:  Function FormatModuleVersionInfo( const aFilename : string) : string');
16905:  Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD) : string');
16906:  Function TBGetFileSize( aFile : string; aMultipleOf : Integer) : Integer');
16907:  Function FormatAttribString( aAttr : Integer) : string');
16908:  Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo) : string');
16909:  Function ShellRecycle( aWnd : HWND; aFileOrFolder : string) : Boolean');
16910:  Function IsDebuggerPresent : BOOL');
16911:  Function TBNotImplemented : HRESULT');
16912: end;
16913:
16914: procedure SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
16915: begin
16916:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16917:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16918:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16919:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16920:   //TDrivesProperty = array['A'..'Z'] of boolean;
16921:  Function TBSetSystemTime( DateTime : TDateTime; DOW : word) : boolean');
16922:  Function IsElevated : Boolean');
16923:  Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
16924:   CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16925:  Function TrimNetResource( UNC : string) : string');
16926:  Procedure GetFreeDrives( var FreeDrives : TDrivesProperty)');
16927:  Procedure GetMappedDrives( var MappedDrives : TDrivesProperty)');
16928:  Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
       UserUsage : TPasswordUsage; User : string; Comment : string) : boolean');
16929:  Function UnmapDrive( Drive : char; Force : boolean) : boolean');
16930:  Function TBIsWindowsVista : Boolean');
16931:  Procedure SetVistaFonts( const AForm : TForm)');
16932:  Procedure SetVistaContentFonts( const AFont : TFont)');
16933:  Function GetProductType( var sType : String) : Boolean');
16934:  Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16935:  Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16936:  Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar');
16937:  Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
```

```
16938:  Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16939:  Function lstrlen( lpString : PChar) : Integer');
16940:  Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16941:  Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD) : BOOL');
16942: end;
16943:
16944: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16945: begin
16946:  'cLineTerminator','Char').SetString( #10);
16947:  'cLineTerminators','String').SetString( #13#10);
16948:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
16949:   SIRegister_TFixedCriticalSection(CL);
16950:   SIRegister_TMultiReadSingleWrite(CL);
16951:  CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16952:  Function GetDecimalSeparator : Char');
16953:  TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16954:  Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
       recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16955:   CL.AddTypeS('NativeInt', 'Integer');
16956:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16957:   CL.AddTypeS('NativeUInt', 'Cardinal');
16958:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16959:   //CL.AddTypeS('TBytes', 'array of Byte');
16960:   CL.AddTypeS('RawByteString', 'UnicodeString');
16961:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16962:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16963:   SIRegister_TPath(CL);
16964:   SIRegister_TFile(CL);
16965:   SIRegister_TdwsThread(CL);
16966:  Function GetSystemMilliseconds : Int64');
16967:  Function UTCDateTime : TDateTime');
16968:  Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16969:  Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16970:  Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16971:  Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
16972:  Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer;');
16973:  Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer;');
16974:  Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16975:  Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16976:  Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16977:  Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16978:  Function InterlockedIncrement( var val : Integer) : Integer');
16979:  Function InterlockedDecrement( var val : Integer) : Integer');
16980:  Procedure FastInterlockedIncrement( var val : Integer)');
16981:  Procedure FastInterlockedDecrement( var val : Integer)');
16982:  Function InterlockedExchangePointer( var target : ___Pointer; val : ___Pointer) : ___Pointer');
16983:  Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16984:  Procedure dwsOutputDebugString( const msg : UnicodeString)');
16985:  Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16986:  Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16987:  Procedure VarCopy( out dest : Variant; const src : Variant)');
16988:  Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16989:  Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16990:  Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16991:  Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16992:  Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
16993:  Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
16994:  Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
16995:  Procedure CloseFileHandle( hFile : THandle)');
16996:  Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
16997:  Function FileMove( const existing, new : UnicodeString) : Boolean');
16998:  Function dwsFileDelete( const fileName : String) : Boolean');
16999:  Function FileRename( const oldName, newName : String) : Boolean');
17000:  Function dwsFileSize( const name : String) : Int64');
17001:  Function dwsFileDateTime( const name : String) : TDateTime');
17002:  Function DirectSet8087CW( newValue : Word) : Word');
17003:  Function DirectSetMXCSR( newValue : Word) : Word');
17004:  Function TtoObject( const T: byte) : TObject');
17005:  Function TtoPointer( const T: byte) : ___Pointer');
17006:  Procedure GetMemForT(var T: byte; Size : integer)');
17007:  Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
17008: end;
17009:
17010: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
17011: begin
17012:  'IPStrSize','LongInt').SetInt( 15);
17013:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
17014:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
17015:  'ADWSBASE','LongInt').SetInt( 9000);
17016:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
17017:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
17018:   SIRegister_EApdSocketException(CL);
17019:   TWsMode', '( wsClient, wsServer )');
17020:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket');
17021:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrCode : Integer)');
17022:   SIRegister_TApdSocket(CL);
17023: end;
```

```
17024:
17025: procedure SIRegister_AdPort(CL: TPSPascalCompiler);
17026: begin
17027:   SIRegister_TApdCustomComPort(CL);
17028:   SIRegister_TApdComPort(CL);
17029:  Function ComName( const ComNumber : Word) : ShortString');
17030:  Function SearchComPort( const C : TComponent) : TApdCustomComPort');
17031: end;
17032:
17033: procedure SIRegister_PathFunc(CL: TPSPascalCompiler);
17034: begin
17035:  Function inAddBackslash( const S : String) : String');
17036:  Function PathChangeExt( const Filename, Extension : String) : String');
17037:  Function PathCharCompare( const S1, S2 : PChar) : Boolean');
17038:  Function PathCharIsSlash( const C : Char) : Boolean');
17039:  Function PathCharIsTrailByte( const S : String; const Index : Integer) : Boolean');
17040:  Function PathCharLength( const S : String; const Index : Integer) : Integer');
17041:  Function inPathCombine( const Dir, Filename : String) : String');
17042:  Function PathCompare( const S1, S2 : String) : Integer');
17043:  Function PathDrivePartLength( const Filename : String) : Integer');
17044:  Function PathDrivePartLengthEx(const Filename:String;const IncludeSignificantSlash:Bool):Int;
17045:  Function inPathExpand( const Filename : String) : String');
17046:  Function PathExtensionPos( const Filename : String) : Integer');
17047:  Function PathExtractDir( const Filename : String) : String');
17048:  Function PathExtractDrive( const Filename : String) : String');
17049:  Function PathExtractExt( const Filename : String) : String');
17050:  Function PathExtractName( const Filename : String) : String');
17051:  Function PathExtractPath( const Filename : String) : String');
17052:  Function PathIsRooted( const Filename : String) : Boolean');
17053:  Function PathLastChar( const S : String) : PChar');
17054:  Function PathLastDelimiter( const Delimiters, S : string) : Integer');
17055:  Function PathLowercase( const S : String) : String');
17056:  Function PathNormalizeSlashes( const S : String) : String');
17057:  Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
17058:  Function PathPos( Ch : Char; const S : String) : Integer');
17059:  Function PathStartsWith( const S, AStartsWith : String) : Boolean');
17060:  Function PathStrNextChar( const S : PChar) : PChar');
17061:  Function PathStrPrevChar( const Start, Current : PChar) : PChar');
17062:  Function PathStrScan( const S : PChar; const C : Char) : PChar');
17063:  Function inRemoveBackslash( const S : String) : String');
17064:  Function RemoveBackslashUnlessRoot( const S : String) : String');
17065:  Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean)');
17066: end;
17067:
17068:
17069: procedure SIRegister_CmnFunc2(CL: TPSPascalCompiler);
17070: begin
17071:  NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion');
17072:  NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer');
17073:  NEWREGSTR_PATH_SPECIAL_FOLDERS','String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders');
17074:  NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall');
17075:  NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName');
17076:  NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString');
17077:  KEY_WOW64_64KEY','LongWord').SetUInt( $0100);
17078:   //CL.AddTypeS('PLeadByteSet', '^TLeadByteSet // will not work');
17079:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17080:   SIRegister_TOneShotTimer(CL);
17081:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17082:  'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17083:  Function NewFileExists( const Name : String) : Boolean');
17084:  Function inDirExists( const Name : String) : Boolean');
17085:  Function FileOrDirExists( const Name : String) : Boolean');
17086:  Function IsDirectoryAndNotReparsePoint( const Name : String) : Boolean');
17087:  Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String');
17088:  Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filenam:String):Longint;
17089:  Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filename:String): Boolean');
17090:  Function IniKeyExists( const Section, Key, Filename : String) : Boolean');
17091:  Function IsIniSectionEmpty( const Section, Filename : String) : Boolean');
17092:  Function SetIniString( const Section, Key, Value, Filename : String) : Boolean');
17093:  Function SetIniInt(const Section,Key:String;const Value: Longint;const Filename: String):Boolean');
17094:  Function SetIniBool(const Section,Key:String;const Value: Boolean;const Filename: String):Boolean');
17095:  Procedure DeleteIniEntry( const Section, Key, Filename : String)');
17096:  Procedure DeleteIniSection( const Section, Filename : String)');
17097:  Function GetEnv( const EnvVar : String) : String');
17098:  Function GetCmdTail : String');
17099:  Function GetCmdTailEx( StartIndex : Integer) : String');
17100:  Function NewParamCount : Integer');
17101:  Function NewParamStr( Index : Integer) : string');
17102:  Function AddQuotes( const S : String) : String');
17103:  Function RemoveQuotes( const S : String) : String');
17104:  Function inGetShortName( const LongName : String) : String');
17105:  Function inGetWinDir : String');
17106:  Function inGetSystemDir : String');
17107:  Function GetSysWow64Dir : String');
17108:  Function GetSysNativeDir( const IsWin64 : Boolean) : String');
17109:  Function inGetTempDir : String');
17110:  Function StringChange( var S : String; const FromStr, ToStr : String) : Integer');
17111:  Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17112:  Function AdjustLength( var S : String; const Res : Cardinal) : Boolean');
```

```
17113:  Function UsingWinNT : Boolean');
17114:  Function ConvertConstPercentStr( var S : String) : Boolean');
17115:  Function ConvertPercentStr( var S : String) : Boolean');
17116:  Function ConstPos( const Ch : Char; const S : String) : Integer');
17117:  Function SkipPastConst( const S : String; const Start : Integer) : Integer');
17118:  Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String) : Boolean');
17119:  Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String):Boolean;
17120:  Function RegValueExists( H : HKEY; Name : PChar) : Boolean');
17121:  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;
17122:  Function RegOpenKeyExView(const
        RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult:HKEY):Longint;
17123:  Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar) : Longint;
17124:  Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17125:  Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubkeyName:PChar):Longint;
17126:  Function GetShellFolderPath( const FolderID : Integer) : String');
17127:  Function IsAdminLoggedOn : Boolean');
17128:  Function IsPowerUserLoggedOn : Boolean');
17129:  Function IsMultiByteString( const S : AnsiString) : Boolean');
17130:  Function FontExists( const FaceName : String) : Boolean');
17131:  //Procedure FreeAndNil( var Obj)');
17132:  Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT) : HMODULE');
17133:  Function GetUILanguage : LANGID');
17134:  Function RemoveAccelChar( const S : String) : String');
17135:  Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17136:  Function AddPeriod( const S : String) : String');
17137:  Function GetExceptMessage : String');
17138:  Function GetPreferredUIFont : String');
17139:  Function IsWildcard( const Pattern : String) : Boolean');
17140:  Function WildcardMatch( const Text, Pattern : PChar) : Boolean');
17141:  Function IntMax( const A, B : Integer) : Integer');
17142:  Function Win32ErrorString( ErrorCode : Integer) : String');
17143:  Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet)');
17144:  Function inCompareMem( P1, P2 : TObject; Length : Integer) : Boolean');
17145:  Function DeleteDirTree( const Dir : String) : Boolean');
17146:  Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17147:  Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT)');
17148:  // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17149:  Function inCharInSet( C : Char; const CharSet : TSysCharSet) : Boolean');
17150:  Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String) : Boolean');
17151:  Function ShutdownBlockReasonDestroy( Wnd : HWND) : Boolean');
17152:  Function TryStrToBoolean( const S : String; var BoolResult : Boolean) : Boolean');
17153:  Procedure WaitMessageWithTimeout( const Milliseconds : DWORD)');
17154:  Function MoveFileReplace(const ExistingFileName, NewFileName : String) : Boolean');
17155:  Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND)');
17156: end;
17157:
17158: procedure SIRegister_CmnFunc(CL: TPSPascalCompiler);
17159: begin
17160:   SIRegister_TWindowDisabler(CL);
17161:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17162:   TMsgBoxCallbackFunc','procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17163:  Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox)');
17164:  Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17165:  Function AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Integer');
17166:  Function MsgBoxP( const Text,Caption: PChar; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17167:  Function inMsgBox(const Text,Caption:String; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17168:  Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
        Typ:TMsgBoxType;const Buttons:Cardinal):Integer');
17169:  Procedure ReactivateTopWindow');
17170:  Procedure SetMessageBoxCaption( const Typ : TMsgBoxType; const NewCaption : PChar)');
17171:  Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean)');
17172:  Procedure SetMessageBoxCallbackFunc(const AFunc : TMsgBoxCallbackFunc; const AParam : LongInt)');
17173: end;
17174:
17175: procedure SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17176: begin
17177:   SIRegister_TImageGrabber(CL);
17178:   SIRegister_TCaptureDrivers(CL);
17179:   SIRegister_TCaptureDriver(CL);
17180: end;
17181:
17182: procedure SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17183: begin
17184:  Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
        Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean');
17185:  Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
        Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean');
17186: end;
17187:
17188: procedure SIRegister_RedirFunc(CL: TPSPascalCompiler);
17189: begin
17190:   CL.AddTypeS('TPreviousFsRedirectionState', record DidDisable : Boolean; OldValue : ___Pointer; end');
17191:  CL.AddDelphiFunction('Function AreFsRedirectionFunctionsAvailable : Boolean');
17192:  Function DisableFsRedirectionIf(const Disable:Boolean;var PreviousState:TPreviousFsRedirectionState):Boo;
17193:  Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17194:  Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17195:  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;
```

```
17196:  Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
       FailIfExists : BOOL) : BOOL');
17197:  Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17198:  Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17199:  Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17200:  Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
       TWin32FindData) : THandle');
17201:  Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD');
17202:  Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String) : String');
17203:  Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
       TFileVersionNumbers) : Boolean');
17204:  Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17205:  Function MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
17206:  Function MoveFileExRedir(const DisableFsRedir:Bool;const ExistingFilen,NewFilename:String;const
       Flags:DWORD):BOOL;
17207:  Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17208:  Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17209:  Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17210:  Function SetNTFSCompressionRedir(const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17211:   SIRegister_TFileRedir(CL);
17212:   SIRegister_TTextFileReaderRedir(CL);
17213:   SIRegister_TTextFileWriterRedir(CL);
17214: end;
17215:
17216: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17217: begin
17218:   //CL.AddTypeS('LongWord', 'Cardinal');
17219:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17220:  Function Compare64( const N1, N2 : Integer64) : Integer');
17221:  Procedure Dec64( var X : Integer64; N : LongWord)');
17222:  Procedure Dec6464( var X : Integer64; const N : Integer64)');
17223:  Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord');
17224:  Function Inc64( var X : Integer64; N : LongWord) : Boolean');
17225:  Function Inc6464( var X : Integer64; const N : Integer64) : Boolean');
17226:  Function Integer64ToStr( X : Integer64) : String');
17227:  Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord');
17228:  Function Mul64( var X : Integer64; N : LongWord) : Boolean');
17229:  Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)');
17230:  Procedure Shr64( var X : Integer64; Count : LongWord)');
17231:  Function StrToInteger64( const S : String; var X : Integer64) : Boolean');
17232: end;
17233:
17234: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17235: begin
17236:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17237:   SIRegister_TSimpleStringList(CL);
17238:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle)');
17239:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17240:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17241:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17242:  // TMD5Digest = array[0..15] of Byte;
17243:   //     TSHA1Digest = array[0..19] of Byte;
17244:  Function CheckForMutexes( Mutexes : String) : Boolean');
17245:  Function CreateTempDir : String');
17246:  Function DecrementSharedCount( const RegView : TRegView; const Filename : String) : Boolean');
17247:  Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
       FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)');
17248:  //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
       DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
       TDeleteFileProc; const Param : Pointer) : Boolean');
17249:  //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
       TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
       TDetermineDefaultLanguageResult');
17250:  //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17251:  Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean');
17252:  Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17253:  Function GetComputerNameString : String');
17254:  Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17255:  Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String) : TMD5Digest');
17256:  Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest');
17257: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest');
17258:  Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest');
17259:  Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest');
17260: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest');
17261:  Function GetRegRootKeyName( const RootKey : HKEY) : String');
17262:  Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
       TotBytes:Int64):Bool;
17263:  Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
       TotalBytes: Integer64):Bool;
17264:  Function GetUserNameString : String');
17265:  Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17266:  //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
       String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
       ResultCode:Integer) : Boolean');
17267: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
       TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17268:  Procedure InternalError( const Id : String)');
17269:  Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17270:  Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
```

```
17271:  Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17272:  Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17273:  Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17274:  Procedure RaiseFunctionFailedError( const FunctionName : String)');
17275:  Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17276:  Procedure RefreshEnvironment');
17277:  Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17278:  Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17279:  Procedure UnregisterFont( const FontName, FontFilename : String)');
17280:  Function RestartComputer : Boolean');
17281:  Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17282:  Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)');
17283:  Procedure Win32ErrorMsg( const FunctionName : String)');
17284:  Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17285:  Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17286:  //from Func2
17287:  //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
       IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean; '
17288:   //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String');
17289:  Procedure RegisterTypeLibrary( const Filename : String)');
17290:  //Procedure UnregisterTypeLibrary( const Filename : String)');
17291:  //Function UnpinShellLink( const Filename : String) : Boolean');}
17292:  function getVersionInfoEx3: TOSVersionInfoEx;');
17293:  Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17294:  procedure InitOle;');
17295:  Function ExpandConst( const S : String) : String');
17296:  Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17297:  Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17298:  Function ExpandConstIfPrefixed( const S : String) : String');
17299:  Procedure LogWindowsVersion');
17300:  Function EvalCheck( const Expression : String) : Boolean');
17301: end;
17302:
17303: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17304: begin
17305:   CL.AddClassN(CL.FindClass('TOBJECT'),'TResourceDetails');
17306:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17307:   SIRegister_TResourceModule(CL);
17308:   SIRegister_TResourceDetails(CL);
17309:   SIRegister_TAnsiResourceDetails(CL);
17310:   SIRegister_TUnicodeResourceDetails(CL);
17311:  Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17312:  Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17313:  Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17314:  Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17315:  Function ResourceNameToInt( const s : string) : Integer');
17316:  Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17317: end;
17318:
17319:
17320: procedure SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17321: begin
17322:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17323:   with CL.AddClassN(CL.FindClass('TObject'),'TSimpleComPort') do begin
17324:     RegisterMethod('Constructor Create');
17325:     RegisterMethod('Procedure Free');
17326:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17327:     RegisterMethod('Procedure WriteString( const S : String)');
17328:     RegisterMethod('Procedure ReadString( var S : String)');
17329:   end;
17330:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17331:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17332:       SimpleComPort.WriteString(AsciiChar);
17333: end;
17334:
17335:
17336: procedure SIRegister_Console(CL: TPSPascalCompiler);
17337: begin
17338:  CL.AddConstantN('White','LongInt').SetInt( 15); }
17339: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17340:  ('conBW40','LongInt').SetInt( 0);
17341:  ('conCO40','LongInt').SetInt( 1);
17342:  ('conBW80','LongInt').SetInt( 2);
17343:  ('conCO80','LongInt').SetInt( 3);
17344:  ('conMono','LongInt').SetInt( 7);
17345:  CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17346:  //CL.AddConstantN('C40','').SetString( CO40);
17347:  //CL.AddConstantN('C80','').SetString( CO80);
17348:  Function conReadKey : Char');
17349:  Function conKeyPressed : Boolean');
17350:  Procedure conGotoXY( X, Y : Smallint)');
17351:  Function conWhereX : Integer');
17352:  Function conWhereY : Integer');
17353:  Procedure conTextColor( Color : Byte);');
17354:  Function conTextColor1 : Byte;');
17355:  Procedure conTextBackground( Color : Byte);');
17356:  Function conTextBackground1 : Byte;');
17357:  Procedure conTextMode( Mode : Word)');
17358:  Procedure conLowVideo');
```

```
17359:  Procedure conHighVideo');
17360:  Procedure conNormVideo');
17361:  Procedure conClrScr');
17362:  Procedure conClrEol');
17363:  Procedure conInsLine');
17364:  Procedure conDelLine');
17365:  Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17366:  Function conScreenWidth : Smallint');
17367:  Function conScreenHeight : Smallint');
17368:  Function conBufferWidth : Smallint');
17369:  Function conBufferHeight : Smallint');
17370:  procedure InitScreenMode;');
17371: end;
17372:
17373: (*----------------------------------------------------------------------------*)
17374: procedure SIRegister_testutils(CL: TPSPascalCompiler);
17375: begin
17376:   SIRegister_TNoRefCountObject(CL);
17377:  Procedure FreeObjects( List : TFPList)');
17378:  Procedure GetMethodList( AObject : TObject; AList : TStrings);');
17379:  Procedure GetMethodList1( AClass : TClass; AList : TStrings);');
17380: end;
17381:
17382: procedure SIRegister_ToolsUnit(CL: TPSPascalCompiler);
17383: begin
17384:  'MaxDataSet','LongInt').SetInt( 35);
17385:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17386:   SIRegister_TDBConnector(CL);
17387:   SIRegister_TDBBasicsTestSetup(CL);
17388:   SIRegister_TTestDataLink(CL);
17389:  'testValuesCount','LongInt').SetInt( 25);
17390:  Procedure InitialiseDBConnector');
17391:  Procedure FreeDBConnector');
17392:  Function DateTimeToTimeString( d : tdatetime) : string');
17393:  Function TimeStringToDateTime( d : String) : TDateTime');
17394: end;
17395:
17396: procedure SIRegister_fpcunit(CL: TPSPascalCompiler);
17397: begin
17398:   SIRegister_EAssertionFailedError(CL);
17399:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17400:   CL.AddTypeS('TRunMethod', 'Procedure');
17401:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTestResult');
17402:   SIRegister_TTest(CL);
17403:   SIRegister_TAssert(CL);
17404:   SIRegister_TTestFailure(CL);
17405:   SIRegister_ITestListener(CL);
17406:   SIRegister_TTestCase(CL);
17407:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17408:   SIRegister_TTestSuite(CL);
17409:   SIRegister_TTestResult(CL);
17410:  Function ComparisonMsg( const aExpected : string; const aActual : string) : string');
17411: end;
17412:
17413: procedure SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
17414: begin
17415:   TOBJECT'),'ETCPBuffer');
17416:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Intege'
17417:    +r; Head : Integer; Used : Integer; end');
17418:  'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
17419:  'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
17420:  'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
17421:  'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
17422:  Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int;
17423:  Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer)');
17424:  Procedure TCPBufferPack( var TCPBuf : TTCPBuffer)');
17425:  Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer)');
17426:  Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer)');
17427:  Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer)');
17428:  Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer) : Pointer');
17429:  Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer)');
17430:  Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Integer');
17431:  Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf: string; const Size:Integer): Integer');
17432:  Function TCPBufferRemove(var TCPBuf : TTCPBuffer;var Buf: string; const Size:Integer): Integer');
17433:  Procedure TCPBufferClear( var TCPBuf : TTCPBuffer)');
17434:  Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer) : Integer');
17435:  Function TCPBufferUsed( const TCPBuf : TTCPBuffer) : Integer');
17436:  Function TCPBufferEmpty( const TCPBuf : TTCPBuffer) : Boolean');
17437:  Function TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Integer');
17438:  Function TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer');
17439:  Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer)');
17440: end;
17441:
17442: procedure SIRegister_Glut(CL: TPSPascalCompiler);
17443: begin
17444:   //CL.AddTypeS('PInteger', '^Integer // will not work');
17445:   //CL.AddTypeS('PPChar', '^PChar // will not work');
17446:  CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3);
17447:  CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12);
```

```
17448:  CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0);
17449:  CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5);
17450:  CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6);
17451:  Procedure LoadGlut( const dll : String)');
17452:  Procedure FreeGlut');
17453: end;
17454:
17455: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17456: begin
17457:   CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17458:  Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean) : THandle');
17459: end;
17460:
17461: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17462: begin
17463:   TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )');
17464:   TTLedState', '( LedOn, LedOff, LedDisabled )');
17465:   SIRegister_TSwitchLed(CL);
17466:  //CL.AddDelphiFunction('Procedure Register');
17467: end;
17468:
17469: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17470: begin
17471:   CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17472:    +'xisting, fdOpenAlways, fdTruncateExisting )');
17473:   CL.AddTypeS('TIFileAccess', '( faRead, faWrite, faReadWrite )');
17474:   CL.AddTypeS('TIFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17475:   SIRegister_TCustomFile(CL);
17476:   SIRegister_TIFile(CL);
17477:   SIRegister_TMemoryFile(CL);
17478:   SIRegister_TTextFileReader(CL);
17479:   SIRegister_TTextFileWriter(CL);
17480:   SIRegister_TFileMapping(CL);
17481:   SIRegister_EFileError(CL);
17482: end;
17483:
17484: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17485: begin
17486:   CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17487:    +', ffaDirectory, ffaArchive, ffaAnyFile )');
17488:   CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17489:   SIRegister_TFileSearch(CL);
17490: end;
17491:
17492: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17493: begin
17494:  TRGBType', 'record RedHex : string; GreenHex : string; BlueHex :'
17495:    + string; Red : integer; Green : integer; Blue : integer; end');
17496:  Function FadeColor( aColor : Longint; aFade : integer) : Tcolor');
17497:  Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17498: end;
17499:
17500: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17501: begin
17502:  Procedure SaveOscSettings');
17503:  Procedure GetOscSettings');
17504: end;
17505:
17506: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17507: begin
17508:   TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer)');
17509:   RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17510:    FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17511:    64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17512:    Cardinal; EnterCount : Integer; ExitCount : Integer; end');
17513:   SIRegister_TcyDebug(CL);
17514: end;
17515:
17516: (*----------------------------------------------------------------------------*)
17517: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17518: begin
17519:   TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )');
17520:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )');
17521:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )');
17522:   SIRegister_TDestinationOptions(CL);
17523:   TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult)');
17524:   TProcOnCopyFileProgressEvent','Procedure(Sender:TObject;FileBytes,
      TransferedBytes:int64;PercentDone:Int64);
17525:   TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String)');
17526:   SIRegister_TcyCopyFiles(CL);
17527:  Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
      ResetAttr:boolean): TCopyFileResult');
17528:  Function cyCopyFileEx( FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
      TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17529:  Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
       ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;'
17530:                       + FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17531: end;
17532:
```

```
17533: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17534: begin
17535:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17536:   SIRegister_TcyFileAttributes(CL);
17537:   SIRegister_TSearchRecInstance(CL);
17538:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17539:   TOptions', 'set of TOption');
17540:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17541:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
      Accept:bool;
17542:   TProcOnValidateDirectoryEvent','Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17543:   SIRegister_TcyCustomSearchFiles(CL);
17544:   SIRegister_TcySearchFiles(CL);
17545:  Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17546:  Function IscyFolder( aSRec : TSearchrec) : Boolean');
17547: end;
17548:
17549: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
17550: begin
17551:  Function jCountChar( const s : string; ch : char) : integer');
17552:  Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings)');
17553:  Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17554:  Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17555:  Function jNormalizeDateTime(const Value:string;theValue:TDateTime;const theFormat:string):string');
17556:  Function jNormalizeDateSeparator( const s : string) : string');
17557:  Function jIsValidDateString( const Value : string) : boolean');
17558:  Function jIsValidTimeString( const Value : string) : boolean');
17559:  Function jIsValidDateTimeString( const Value : string) : boolean');
17560: end;
17561:
17562: procedure SIRegister_kcMapViewer(CL: TPSPascalCompiler);
17563: begin
17564:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapViewer');
17565:   CL.AddTypeS('TMapSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
17566:    +'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
17567:    +', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
17568:    +'EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
17569:    +'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17570:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17571:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17572:   TIntPoint', 'record X : Int64; Y : Int64; end');
17573:   TkcRealPoint', 'record X : Extended; Y : Extended; end');
17574:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17575:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17576:   SIRegister_TCustomDownloadEngine(CL);
17577:   SIRegister_TCustomGeolocationEngine(CL);
17578:   SIRegister_TMapViewer(CL);
17579: end;
17580:
17581: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17582: begin
17583:  (*CL.AddDelphiFunction('Function isFunc( name : TNamePart) : Boolean');*)
17584:  CL.AddDelphiFunction('Function isUnnamedFunc( name : TNamepart) : Boolean');
17585:  Function isPtrToFunc( name : TNamePart) : Boolean');
17586:  Function isFuncRetFuncPtr( name : TNamePart) : Boolean');
17587:  Function isPtrToFuncRetFuncPtr( name : TNamePart) : Boolean');
17588:  Function GetFuncParam( name : TNamePart) : TNamePart');
17589:  Function isArray( name : TNamePart) : Boolean');
17590:  Function GetArrayPart( name : TNamePart) : TNamePart');
17591:  Function GetIdFromPart( name : TNamePart) : AnsiString');
17592:  Function GetIdPart( name : TNamePart) : TNamePart');
17593:  Function isNamePartPtrToFunc( part : TNamePart) : Boolean');
17594:  Function isAnyBlock( part : TNamePart) : Boolean');*)
17595:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17596:   SIRegister_TLineBreaker(CL);
17597:   CL.AddTypeS('TNameKind', 'Integer');
17598:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17599:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17600:  Function SphericalMod( X : Extended) : Extended');
17601:  Function cSign( Value : Extended) : Extended');
17602:  Function LimitFloat( const eValue, eMin, eMax : Extended) : Extended');
17603:  Function AngleToRadians( iAngle : Extended) : Extended');
17604:  Function RadiansToAngle( eRad : Extended) : Extended');
17605:  Function Cross180( iLong : Double) : Boolean');
17606:  Function Mod180( Value : integer) : Integer');
17607:  Function Mod180Float( Value : Extended) : Extended');
17608:  Function MulDivFloat( a, b, d : Extended) : Extended');
17609:  Function LongDiff( iLong1, iLong2 : Double) : Double');
17610:  Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap)');
17611:  Function Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap');
17612:  Function FixFilePath( const Inpath, CheckPath : string) : string');
17613:  Function UnFixFilePath( const Inpath, CheckPath : string) : string');
17614:  Procedure FillStringList( sl : TStringList; const aText : string)');
17615: end;
17616:
17617: procedure SIRegister_LedNumber(CL: TPSPascalCompiler);
17618: begin
17619:   TLedSegmentSize', 'Integer');
17620:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');
```

```
17621:   SIRegister_TCustomLEDNumber(CL);
17622:   SIRegister_TLEDNumber(CL);
17623: end;
17624:
17625: procedure SIRegister_StStrL(CL: TPSPascalCompiler);
17626: begin
17627:   CL.AddTypeS('LStrRec', 'record AllocSize : Longint; RefCount : Longint; Length : Longint; end');
17628:    CL.AddTypeS('AnsiChar', 'Char');
17629:     CL.AddTypeS('BTable', 'array[0..255] of Byte');       //!!!
17630:   CL.AddDelphiFunction('Function HexBL( B : Byte) : AnsiString');
17631:  Function HexWL( W : Word) : AnsiString');
17632:  Function HexLL( L : LongInt) : AnsiString');
17633:  Function HexPtrL( P : ___Pointer) : AnsiString');
17634:  Function BinaryBL( B : Byte) : AnsiString');
17635:  Function BinaryWL( W : Word) : AnsiString');
17636:  Function BinaryLL( L : LongInt) : AnsiString');
17637:  Function OctalBL( B : Byte) : AnsiString');
17638:  Function OctalWL( W : Word) : AnsiString');
17639:  Function OctalLL( L : LongInt) : AnsiString');
17640:  Function Str2Int16L( const S : AnsiString; var I : SmallInt) : Boolean');
17641:  Function Str2WordL( const S : AnsiString; var I : Word) : Boolean');
17642:  Function Str2LongL( const S : AnsiString; var I : LongInt) : Boolean');
17643:  Function Str2RealL( const S : AnsiString; var R : Double) : Boolean');
17644:  Function Str2RealL( const S : AnsiString; var R : Real) : Boolean');
17645:  Function Str2ExtL( const S : AnsiString; var R : Extended) : Boolean');
17646:  Function Long2StrL( L : LongInt) : AnsiString');
17647:  Function Real2StrL( R : Double; Width : Byte; Places : ShortInt) : AnsiString');
17648:  Function Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : AnsiString');
17649:  Function ValPrepL( const S : AnsiString) : AnsiString');
17650:  Function CharStrL( C : Char; Len : Cardinal) : AnsiString');
17651:  Function PadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17652:  Function PadLL( const S : AnsiString; Len : Cardinal) : AnsiString');
17653:  Function LeftPadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17654:  Function LeftPadL( const S : AnsiString; Len : Cardinal) : AnsiString');
17655:  Function TrimLeadL( const S : AnsiString) : AnsiString');
17656:  Function TrimTrailL( const S : AnsiString) : AnsiString');
17657:  Function TrimL( const S : AnsiString) : AnsiString');
17658:  Function TrimSpacesL( const S : AnsiString) : AnsiString');
17659:  Function CenterChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17660:  Function CenterL( const S : AnsiString; Len : Cardinal) : AnsiString');
17661:  Function EntabL( const S : AnsiString; TabSize : Byte) : AnsiString');
17662:  Function DetabL( const S : AnsiString; TabSize : Byte) : AnsiString');
17663:  Function ScrambleL( const S, Key : AnsiString) : AnsiString');
17664:  Function SubstituteL( const S, FromStr, ToStr : AnsiString) : AnsiString');
17665:  Function FilterL( const S, Filters : AnsiString) : AnsiString');
17666:  Function CharExistsL( const S : AnsiString; C : AnsiChar) : Boolean');
17667:  Function CharCountL( const S : AnsiString; C : AnsiChar) : Cardinal');
17668:  Function WordCountL( const S, WordDelims : AnsiString) : Cardinal');
17669:  Function WordPositionL( N : Cardinal; const S, WordDelims : AnsiString; var Pos : Cardinal) : Boolean');
17670:  Function ExtractWordL( N : Cardinal; const S, WordDelims : AnsiString) : AnsiString');
17671:  Function AsciiCountL( const S, WordDelims : AnsiString; Quote : AnsiChar) : Cardinal');
17672:  Function AsciiPositionL(N: Cardinal;const S,WordDelims:AnsiString;Quote:AnsiChar;var Pos:Cardinal):Bool;
17673:  Function ExtractAsciiL( N : Cardinal; const S, WordDelims : AnsiString; Quote : AnsiChar) : AnsiString');
17674:  Procedure WordWrapL(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17675:  Procedure WordWrap(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17676:  Function CompStringL( const S1, S2 : AnsiString) : Integer');
17677:  Function CompUCStringL( const S1, S2 : AnsiString) : Integer');
17678:  Function SoundexL( const S : AnsiString) : AnsiString');
17679:  Function MakeLetterSetL( const S : AnsiString) : Longint');
17680:  Procedure BMMakeTableL( const MatchString : AnsiString; var BT : BTable)');
17681:  Function BMSearchL(var Buffer,BufLength:Cardinal;var BT:BTable;const MatchString:AnsiString;var
       Pos:Card):Bool;
17682:  Function BMSearchUCL( var Buffer, BufLength : Cardinal; var BT : BTable; const MatchString : AnsiString;
       var Pos : Cardinal) : Boolean');
17683:  Function DefaultExtensionL( const Name, Ext : AnsiString) : AnsiString');
17684:  Function ForceExtensionL( const Name, Ext : AnsiString) : AnsiString');
17685:  Function JustFilenameL( const PathName : AnsiString) : AnsiString');
17686:  Function JustNameL( const PathName : AnsiString) : AnsiString');
17687:  Function JustExtensionL( const Name : AnsiString) : AnsiString');
17688:  Function JustPathnameL( const PathName : AnsiString) : AnsiString');
17689:  Function AddBackSlashL( const DirName : AnsiString) : AnsiString');
17690:  Function CleanPathNameL( const PathName : AnsiString) : AnsiString');
17691:  Function HasExtensionL( const Name : AnsiString; var DotPos : Cardinal) : Boolean');
17692:  Function CommaizeL( L : LongInt) : AnsiString');
17693:  Function CommaizeChL( L : Longint; Ch : AnsiChar) : AnsiString');
17694:  Function FloatFormL(const Mask:AnsiString;R:TstFloat;const LtCurr,RtCurr:AnsiString;Sep,
       DecPt:Char):AnsiString;
17695:  Function LongIntFormL(const Mask:AnsiString;L:LongInt;const LtCurr,
       RtCurr:AnsiString;Sep:Char):AnsiString);
17696:  Function StrChPosL( const P : AnsiString; C : AnsiChar; var Pos : Cardinal) : Boolean');
17697:  Function StrStPosL( const P, S : AnsiString; var Pos : Cardinal) : Boolean');
17698:  Function StrStCopyL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17699:  Function StrChInsertL( const S : AnsiString; C : AnsiChar; Pos : Cardinal) : AnsiString');
17700:  Function StrStInsertL( const S1, S2 : AnsiString; Pos : Cardinal) : AnsiString');
17701:  Function StrChDeleteL( const S : AnsiString; Pos : Cardinal) : AnsiString');
17702:  Function StrStDeleteL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17703:  Function ContainsOnlyL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17704:  Function ContainsOtherThanL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17705:  Function CopyLeftL( const S : AnsiString; Len : Cardinal) : AnsiString');
```

```
17706:  Function CopyMidL( const S : AnsiString; First, Len : Cardinal) : AnsiString');
17707:  Function CopyRightL( const S : AnsiString; First : Cardinal) : AnsiString');
17708:  Function CopyRightAbsL( const S : AnsiString; NumChars : Cardinal) : AnsiString');
17709:  Function CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Cardinal;var
       SubString:AString):Bool;
17710:  Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
       SubString:AnsiString):Bool;
17711:  Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean) : AnsiString');
17712:  Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
       var SubString : AnsiString) : Boolean');
17713:  Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
       SubString : AnsiString) : Boolean');
17714:  Function DeleteWithinL( const S, Delimiter : AnsiString) : AnsiString');
17715:  Function ExtractTokensL(const S,
       Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Cardinal;
17716:  Function IsChAlphaL( C : AnsiChar) : Boolean');
17717:  Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17718:  Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17719:  Function IsStrAlphaL( const S : AnsiString) : Boolean');
17720:  Function IsStrNumericL( const S, Numbers : AnsiString) : Boolean');
17721:  Function IsStrAlphaNumericL( const S, Numbers : AnsiString) : Boolean');
17722:  Function KeepCharsL( const S, Chars : AnsiString) : AnsiString');
17723:  Function LastWordL( const S, WordDelims, AWord : AnsiString; var Position : Cardinal) : Boolean');
17724:  Function LastWordAbsL( const S, WordDelims : AnsiString; var Position : Cardinal) : Boolean');
17725:  Function LastStringL( const S, AString : AnsiString; var Position : Cardinal) : Boolean');
17726:  Function LeftTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17727:  Function ReplaceWordL(const S, WordDelims,OldWord,NewWord:AnsiString;N:Card;var
       Replacements:Card):AnsiString;
17728:  Function ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:AnsiString;var
       Replacements:Cardinal):AnsiString');
17729:  Function ReplaceStringL(const S,OldString,NewString:AnsiString;N:Cardinal;var
       Replacements:Cardinal):AnsiString;
17730:  Function ReplaceStringAllL(const S,OldString,NewString:AnsiString; var Replacements:Cardinal):AnsiString;
17731:  Function RepeatStringL(const RepeatString:AnsiString;var Repetitions:Cardinal;MaxLen:Cardinal):AnsiString;
17732:  Function RightTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17733:  Function StrWithinL( const S, SearchStr : string; Start : Cardinal; var Position : Cardinal) : boolean');
17734:  Function TrimCharsL( const S, Chars : AnsiString) : AnsiString');
17735:  Function WordPosL(const S,WordDelims,AWord: AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17736:  Function WordPos(const S,WordDelims,AWord:AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17737:  end;
17738:
17739:  procedure SIRegister_pwnative_out(CL: TPSPascalCompiler);
17740: begin
17741:  CL.AddConstantN('STDIN','LongInt').SetInt( 0);
17742:  ('STDOUT','LongInt').SetInt( 1);
17743:  ('STDERR','LongInt').SetInt( 2);
17744:  Procedure NativeWrite( s : astr);');
17745:  Procedure NativeWrite1( PString : PChar);');
17746:  Procedure NativeWrite2( Buffer : PChar; NumChars : Cardinal);');
17747:  Procedure NativeWriteLn( s : astr);');
17748:  Procedure NativeWriteLn1;');
17749: end;
17750:
17751: procedure SIRegister_synwrap1(CL: TPSPascalCompiler);
17752: begin
17753:   CL.AddTypeS(TSynwInfo', 'record Err : byte; UrlHtml : ansistring; ErrResponse
17754:    : integer; UltimateURL : ansistring; Headers : ansistring; end');
17755:   CL.AddTypeS('TUrlInfo', 'record Err : byte; UltimateURL : string; end');
17756:  Function GetHttpFile( const Url, UserAgent, Outfile : string; verbose : boolean): TSynwInfo;');
17757:  Function GetHttpFile1( const Url, UserAgent, Outfile : string) : TSynwInfo;');
17758:  Function GetHttpFile2( const Url, Outfile : string) : TSynwInfo;');
17759:  Function GetHttpFile3( const Url, outfile : string; verbose : boolean) : TSynwInfo;');
17760:  Function GetHtm( const Url : string) : string;');
17761:  Function GetHtm1( const Url, UserAgent : string) : string;');
17762:  Function GetUrl( const Url : string; verbose : boolean) : TSynwInfo;');
17763:  Function GetUrl1( const Url, useragent : string) : TSynwInfo;');
17764:  Function GetUrl2( const Url : string) : TSynwInfo;');
17765:  Function GetUrl3( const Url : string; const http : THTTPSend; verbose : boolean): TUrlInfo;');
17766:  Function GetUrl4( const Url : string; const http : THTTPSend) : TUrlInfo;');
17767:  Procedure StrToStream( s : String; strm : TMemoryStream)');
17768:  Function StrLoadStream( strm : TStream) : String');
17769:
17770: end;
17771:
17772: procedure SIRegister_HTMLUtil(CL: TPSPascalCompiler);
17773: begin
17774:  Function GetVal( const tag, attribname_ci : string) : string');
17775:  Function GetTagName( const Tag : string) : string');
17776:  Function GetUpTagName( const tag : string) : string');
17777:  Function GetNameValPair( const tag, attribname_ci : string) : string');
17778:  Function GetValFromNameVal( const namevalpair : string) : string');
17779:  Function GetNameValPair_cs( const tag, attribname : string) : string');
17780:  Function GetVal_JAMES( const tag, attribname_ci : string) : string');
17781:  Function GetNameValPair_JAMES( const tag, attribname_ci : string) : string');
17782:  Function CopyBuffer( StartIndex : PChar; Len : integer) : string');
17783:  Function Ucase( s : string) : string');
17784: end;
17785:
17786: procedure SIRegister_pwmain(CL: TPSPascalCompiler);
```

```
17787: begin
17788:  CL.AddConstantN('FUTURE_COOKIE','String').SetString( 'Mon, 01 Dec 2099 12:00:00 GMT');
17789:  EXPIRED_COOKIE','String').SetString( 'Mon, 01 Jan 2001 12:00:00 GMT');
17790:  'SECURE_OFF','LongInt').SetInt( 0);
17791:  'SECURE_ON','LongInt').SetInt( 2);
17792:  'SECURE_FILTER','LongInt').SetInt( 3);
17793:   THandle or DWord!
17794:  //  astr = ansistring;
17795:   CL.AddTypeS('pastr', 'ansistring');
17796:   CL.AddTypeS('TFilterFunc', 'function(const s: pastr): pastr;');
17797:   uses pwinit at begin
17798:   //type  TFilterFunc = function(const s: astr): astr;
17799:  Demo: ..\maxbox3\examples2\519_powtils.txt
17800:
17801:  //CL.AddConstantN('CASE_SENSITIVE','Boolean')BoolToStr( false);
17802:  //CL.AddConstantN('CASE_IGNORE','Boolean')BoolToStr( false);
17803:  Procedure pwInit');
17804:  Procedure OffReadln');
17805:  Function Lcase( const s : pastr) : pastr');
17806:  Function Ucase( const s : pastr) : pastr');
17807:  Function CountPostVars : longword');
17808:  Function GetPostVar( const name : pastr) : pastr;');
17809:  Function GetPostVar1( const name : pastr; filter : TFilterFunc) : pastr;');
17810:  Function GetPostVar_S( const name : pastr; Security : integer) : pastr');
17811:  Function GetPostVar_SF( const name : pastr; Security : integer) : pastr');
17812:  Function GetPostVarAsFloat( const name : pastr) : double');
17813:  Function GetPostVarAsInt( const name : pastr) : longint');
17814:  Function GetPostVar_SafeHTML( const name : pastr) : pastr');
17815:  Function FetchPostVarName( idx : longword) : pastr');
17816:  Function FetchPostVarVal( idx : longword) : pastr;');
17817:  Function FetchPostVarVal1( idx : longword; filter : TFilterFunc) : pastr;');
17818:  Function FetchPostVarName_S( idx : longword; Security : integer) : pastr');
17819:  Function FetchPostVarVal_S( idx : longword; Security : integer) : pastr');
17820:  Function IsPostVar( const name : pastr) : boolean');
17821:  Function CountAny : longword');
17822:  Function GetAny( const name : pastr) : pastr;');
17823:  Function GetAny1( const name : pastr; filter : TFilterFunc) : pastr;');
17824:  Function GetAny_S( const name : pastr; Security : integer) : pastr');
17825:  Function GetAnyAsFloat( const name : pastr) : double');
17826:  Function GetAnyAsInt( const name : pastr) : longint');
17827:  Function IsAny( const name : pastr) : byte');
17828:  Function CountCookies : longword');
17829:  Function FetchCookieName( idx : longword) : pastr');
17830:  Function FetchCookieVal( idx : longword) : pastr;');
17831:  Function FetchCookieVal1( idx : longword; filter : TFilterFunc) : pastr;');
17832:  Function GetCookie( const name : pastr) : pastr;');
17833:  Function GetCookie1( const name : pastr; filter : TFilterFunc) : pastr;');
17834:  Function GetCookieAsFloat( const name : pastr) : double');
17835:  Function GetCookieAsInt( const name : pastr) : longint');
17836:  Function IsCookie( const name : pastr) : boolean');
17837:  Function SetCookie( const name, value : pastr) : boolean');
17838:  Function SetCookieAsFloat( const name : pastr; value : double) : boolean');
17839:  Function SetCookieAsInt( const name : pastr; value : longint) : boolean');
17840:  Function SetCookieEx( const name, value, path, domain, expiry : pastr) : boolean');
17841:  Function SetCookieAsFloatEx(const name:pastr;value : double; const path,domain,expiry:pastr):bool;
17842:  Function SetCookieAsIntEx( const name:pastr;value : longint; const path,domain,expiry:pastr):bool;
17843:  Function UnsetCookie( const name : pastr) : boolean');
17844:  Function UnsetCookieEx( const name, path, domain : pastr) : boolean');
17845:  Function FilterHtml( const input : pastr) : pastr');
17846:  Function FilterHtml_S( const input : pastr; security : integer) : pastr');
17847:  Function TrimBadChars( const input : pastr) : pastr');
17848:  Function TrimBadFile( const input : pastr) : pastr');
17849:  Function TrimBadDir( const input : pastr) : pastr');
17850:  Function TrimBad_S( const input : pastr; security : integer) : pastr');
17851:  Function CountHeaders : longword');
17852:  Function FetchHeaderName( idx : longword) : pastr');
17853:  Function FetchHeaderVal( idx : longword) : pastr');
17854:  Function GetHeader( const name : pastr) : pastr');
17855:  Function IsHeader( const name : pastr) : boolean');
17856:  Function SetHeader( const name, value : pastr) : boolean');
17857:  Function UnsetHeader( const name : pastr) : boolean');
17858:  Function PutHeader( const header : pastr) : boolean');
17859:  Procedure Out1( const s : pastr)');
17860:  Procedure OutLn( const s : pastr)');
17861:  Procedure OutA( args : array of const)');
17862:  Procedure OutF( const s : pastr)');
17863:  Procedure OutLnF( const s : pastr)');
17864:  Procedure OutFF( const s : pastr)');
17865:  Procedure OutF_FI( const s : pastr; HTMLFilter : boolean)');
17866:  Procedure OutLnFF( const s : pastr)');
17867:  Procedure OutLnF_FI( const s : pastr; HTMLFilter : boolean)');
17868:  Function FileOut( const fname : pastr) : word');
17869:  Function ResourceOut( const fname : pastr) : word');
17870:  Procedure BufferOut( const buff, len : LongWord)');
17871:  Function TemplateOut( const fname : pastr; HtmlFilter : boolean) : word;');
17872:  Function TemplateOut1( const fname : pastr) : word;');
17873:  Function TemplateOut2( const fname : pastr; filter : TFilterFunc) : word;');
17874:  Function TemplateOut3( const fname : pastr; HtmlFilter : boolean) : word;');
17875:  Function TemplateRaw( const fname : pastr) : word');
```

```
17876:  Function Fmt( const s : pastr) : pastr;');
17877:  Function Fmt1( const s : pastr; filter : TFilterFunc) : pastr;');
17878:  Function FmtFilter( const s : pastr) : pastr');
17879:  Function Fmt_SF(const s:pastr;HTMLFilter:bool;filter:TFilterFunc;FilterSecurity,TrimSecurity:int):pastr;
17880:  Function Fmt_SF1(const s:pastr; HTMLFilter:boolean; FilterSecurity, TrimSecurity : integer) : pastr;');
17881:  Function CountRtiVars : longword');
17882:  Function FetchRtiName( idx : longword) : pastr');
17883:  Function FetchRtiVal( idx : longword) : pastr');
17884:  Function GetRti( const name : pastr) : pastr');
17885:  Function GetRtiAsFloat( const name : pastr) : double');
17886:  Function GetRtiAsInt( const name : pastr) : longint');
17887:  Function IsRti( const name : pastr) : boolean');
17888:  Procedure SetRTI( const name, value : pastr)');
17889:  Function FetchUpfileName( idx : longword) : pastr');
17890:  Function GetUpfileName( const name : pastr) : pastr');
17891:  Function GetUpfileSize( const name : pastr) : longint');
17892:  Function GetUpfileType( const name : pastr) : pastr');
17893:  Function CountUpfiles : longword');
17894:  Function IsUpfile( const name : pastr) : boolean');
17895:  Function SaveUpfile( const name, fname : pastr) : boolean');
17896:  Function CountVars : longword');
17897:  Function FetchVarName( idx : longword) : pastr');
17898:  Function FetchVarVal( idx : longword) : pastr;');
17899:  Function FetchVarVal1( idx : longword; filter : TFilterFunc) : pastr;');
17900:  Function GetVar( const name : pastr) : pastr;');
17901:  Function GetVar1( const name : pastr; filter : TFilterFunc) : pastr;');
17902:  Function GetVar_S( const name : pastr; security : integer) : pastr');
17903:  Function GetVarAsFloat( const name : pastr) : double');
17904:  Function GetVarAsInt( const name : pastr) : longint');
17905:  Procedure SetVar( const name, value : pastr)');
17906:  Procedure SetVarAsFloat( const name : pastr; value : double)');
17907:  Procedure SetVarAsInt( const name : pastr; value : longint)');
17908:  Function IsVar( const name : pastr) : byte');
17909:  Procedure UnsetVar( const name : pastr)');
17910:  Function LineEndToBR( const s : pastr) : pastr');
17911:  Function RandomStr( len : longint) : pastr');
17912:  Function XorCrypt( const s : pastr; key : byte) : pastr');
17913:  Function CountCfgVars : longword');
17914:  Function FetchCfgVarName( idx : longword) : pastr');
17915:  Function FetchCfgVarVal( idx : longword) : pastr');
17916:  Function IsCfgVar( const name : pastr) : boolean');
17917:  Function SetCfgVar( const name, value : pastr) : boolean');
17918:  Function GetCfgVar( const name : pastr) : pastr');
17919:  Procedure ThrowErr( const s : pastr)');
17920:  Procedure ThrowWarn( const s : pastr)');
17921:  Procedure ErrWithHeader( const s : pastr)');
17922:   CL.AddTypeS('TWebVar', 'record name : pastr; value : pastr; end');
17923:   CL.AddTypeS('TWebVars', 'array of TWebVar');
17924:  Function iUpdateWebVar(var webv : TWebVars; const name, value:pastr; upcased:boolean):boolean');
17925:  Function iAddWebCfgVar( const name, value : pastr) : boolean');
17926:  Procedure iAddWebVar( var webv : TWebVars; const name, value : pastr)');
17927:  Procedure iSetRTI( const name, value : pastr)');
17928:  Function iCustomSessUnitSet : boolean');
17929:  Function iCustomCfgUnitSet : boolean');
17930: end;
17931:
17932: procedure SIRegister_W32VersionInfo(CL: TPSPascalCompiler);
17933: begin
17934:   SIRegister_TProjectVersionInfo(CL);
17935:  CL.AddDelphiFunction('Function MSLanguageToHex( const s : string) : string');
17936:  Function MSHexToLanguage( const s : string) : string');
17937:  Function MSCharacterSetToHex( const s : string) : string');
17938:  Function MSHexToCharacterSet( const s : string) : string');
17939:  Function MSLanguages : TStringList');
17940:  Function MSHexLanguages : TStringList');
17941:  Function MSCharacterSets : TStringList');
17942:  Function MSHexCharacterSets : TStringList');
17943: end;
17944:
17945: procedure SIRegister_IpUtils(CL: TPSPascalCompiler);
17946: begin
17947:   TIpHandle', 'Cardinal');
17948:   TIpMD5StateArray', 'array[0..3] of DWORD;');
17949:   CL.AddTypeS('TIpMD5CountArray', 'array[0..1] of DWORD;');
17950:   TIpMD5ByteBuf', 'array[0..63] of Byte;');
17951:   TIpMD5LongBuf', 'array[0..15] of DWORD;');
17952:   TIpMD5Digest', 'array[0..15] of Byte;');
17953:   TIpLineTerminator', '( ltNone, ltCR, ltLF, ltCRLF, ltOther )');
17954:   TIpMD5Context', 'record State : TIpMD5StateArray; Count : TIpMD5'
17955:    +'CountArray; ByteBuf : TIpMD5ByteBuf; end');
17956:   CL.AddClassN(CL.FindClass('TOBJECT'),'EIpBaseException');
17957:   CL.AddClassN(CL.FindClass('TOBJECT'),'EIpAccessException');
17958:   CL.AddClassN(CL.FindClass('TOBJECT'),'EIpHtmlException');
17959:   SIRegister_TIpBaseAccess(CL);
17960:   SIRegister_TIpBasePersistent(CL);
17961:   //TIpComponentClass', 'class of TIpBaseComponent');
17962:   SIRegister_TIpBaseComponent(CL);
17963:   SIRegister_TIpBaseWinControl(CL);
17964:  Function InClassA( Addr : LongInt) : Boolean');
```

```
17965:  Function InClassB( Addr : LongInt) : Boolean');
17966:  Function InClassC( Addr : LongInt) : Boolean');
17967:  Function InClassD( Addr : LongInt) : Boolean');
17968:  Function InMulticast( Addr : LongInt) : Boolean');
17969:  Function IpCharCount( const Buffer, BufSize : DWORD; C : AnsiChar) : DWORD');
17970:  Function IpCompStruct( const S1, S2, Size : Cardinal) : Integer');
17971:  Function IpMaxInt( A, B : Integer) : Integer');
17972:  Function IpMinInt( A, B : Integer) : Integer');
17973:  Procedure IpSafeFree( var Obj: TObject)');
17974:  Function IpShortVersion : string');
17975:  Function IpInternetSumPrim( var Data, DataSize, CurCrc : DWORD) : DWORD');
17976:  Function IpInternetSumOfStream( Stream : TStream; CurCrc : DWORD) : DWORD');
17977:  Function IpInternetSumOfFile( const FileName : string) : DWORD');
17978:  Function MD5SumOfFile( const FileName : string) : string');
17979:  Function MD5SumOfStream( Stream : TStream) : string');
17980:  Function MD5SumOfStreamDigest( Stream : TStream) : TIpMD5Digest');
17981:  Function MD5SumOfString( const S : string) : string');
17982:  Function MD5SumOfStringDigest( const S : string) : TIpMD5Digest');
17983:  Function SafeYield : LongInt');
17984:  Function AllTrimSpaces( Strng : string) : string');
17985:  Function IpCharPos( C : AnsiChar; const S : string) : Integer');
17986:  Function CharPosIdx( C : AnsiChar; const S : string; Idx : Integer) : Integer');
17987:  Function NthCharPos( C : AnsiChar; const S : string; Nth : Integer) : Integer');
17988:  Function RCharPos( C : AnsiChar; const S : string) : Integer');
17989:  Function RCharPosIdx( C : AnsiChar; const S : string; Idx : Integer) : Integer');
17990:  Function RNthCharPos( C : AnsiChar; const S : string; Nth : Integer) : Integer');
17991:  Function IpRPos( const Substr : string; const S : string) : Integer');
17992:  Function IpPosIdx( const SubStr, S : string; Idx : Integer) : Integer');
17993:   ACharSet', 'set of AnsiChar');
17994:   TIpAddrRec', 'record Scheme : string; UserName : string; Password string; Authority : string;
17995:    Port : string; Path : string; Fragment : string; Query : string; QueryDelim : AnsiChar; end');
17996:  Procedure Initialize( var AddrRec : TIpAddrRec)');
17997:  Procedure Finalize( var AddrRec : TIpAddrRec)');
17998:  Function ExtractEntityName( const NamePath : string) : string');
17999:  Function ExtractEntityPath( const NamePath : string) : string');
18000:  Function IpParseURL( const URL : string; var Rslt : TIpAddrRec) : Boolean');
18001:  Function BuildURL( const OldURL, NewURL : string) : string');
18002:  Function PutEscapes( const S : string; EscapeSet : ACharSet) : string');
18003:  Function RemoveEscapes( const S : string; EscapeSet : ACharSet) : string');
18004:  Procedure SplitParams( const Parms : string; Dest : TStrings)');
18005:  Function NetToDOSPath( const PathStr : string) : string');
18006:  Function DOSToNetPath( const PathStr : string) : string');
18007:  Procedure SplitHttpResponse( const S : string; var V, MsgID, Msg : string)');
18008:  Procedure FieldFix( Fields : TStrings)');
18009:  Function AppendSlash( APath : string) : string');
18010:  Function RemoveSlash( APath : string) : string');
18011:  Function GetParentPath( const Path : string) : string');
18012:  Function GetLocalContent( const TheFileName : string) : string');
18013:  Function IPDirExists( Dir : string) : Boolean');
18014:  Function GetTemporaryFile( const Path : string) : string');
18015:  Function GetTemporaryPath : string');
18016:  Function AppendBackSlash( APath : string) : string');
18017:  Function IpRemoveBackSlash( APath : string) : string');
18018:  Function INetDateStrToDateTime( const DateStr : string) : TDateTime');
18019:  Function DateTimeToINetDateTimeStr( DateTime : TDateTime) : string');
18020:  Function IpTimeZoneBias : Integer');
18021:  Procedure SplitCookieFields( const Data : string; Fields : TStrings)');
18022: end;
18023:
18024: procedure SIRegister_LrtPoTools(CL: TPSPascalCompiler);
18025: begin
18026:   CL.AddTypeS('TPOStyle', '( postStandard, postPropName, postFull )');
18027:  Procedure Lrt2Po( const LRTFile : string; POStyle : TPOStyle)');
18028:  Procedure CombinePoFiles( SL : TStrings; const FName : string)');
18029: end;
18030:
18031: procedure SIRegister_GPS(CL: TPSPascalCompiler);
18032: begin
18033:  CL.AddConstantN('MAX_SATS','LongInt').SetInt( 12);
18034:  GPSMSG_START','String').SetString( '$');
18035:  GPSMSG_STOP','String').SetString( '*');
18036:  SEC_BETWEEN_SEG','LongInt').SetInt( 5);
18037:   CL.AddTypeS('TSatellite', 'record Identification : Shortint; Elevation : Shor'
18038:    +'tint; Azimut : Smallint; SignLevel : Smallint; end');
18039:   CL.AddTypeS('TSatellites', 'array[1..12] of TSatellite');
18040:  //TSatellites = array[1..MAX_SATS] of TSatellite;
18041:   TGPSSatEvent', 'Procedure ( Sender : TObject; NbSat, NbSatUse: Shortint; Sats : TSatellites)');
18042:   TGPSDatas', 'record Latitude : Double; Longitude : Double; Heigh'
18043:    tAboveSea : Double; Speed : Double; UTCTime : TDateTime; Valid : Boolean; '
18044:    NbrSats : Shortint; NbrSatsUsed : Shortint; Course : Double; end');
18045:   CL.AddTypeS('TGPSDatasEvent', 'Procedure ( Sender : TObject; GPSDatas : TGPSDatas)');
18046:   CL.AddTypeS('TMsgGP', '( msgGP, msgGPGGA, msgGPGLL, msgGPGSV, msgGPRMA, msgGPRMC, msgGPZDA )');
18047:   CL.AddTypeS('TSpeedUnit', '( suKilometre, suMile, suNauticalMile )');
18048:   SIRegister_TGPSLink(CL);
18049:   SIRegister_TCustomGPS(CL);
18050:   SIRegister_TGPS(CL);
18051:   SIRegister_TGPStoGPX(CL);
18052:   SIRegister_TGPSSpeed(CL);
18053:   SIRegister_TGPSSatellitesPosition(CL);
```

```
18054:   SIRegister_TGPSSatellitesReception(CL);
18055:   SIRegister_TGPSCompass(CL);
18056:  //CL.AddDelphiFunction('Procedure Register( )');
18057:  Function IndexMsgGP( StrMsgGP : String) : TMsgGP');
18058:  Function StrCoordToAngle( Point : Char; Angle : String) : Double');
18059:  Function StrTimeToTime( const Time : String) : TDateTime');
18060:  Function StrToInteger( const Str : String) : Integer');
18061:  Function StrToReal( const Str : String) : Extended');
18062:  Function GPSRotatePoint( Angle : Double; Ct, Pt : TPoint) : TPoint');
18063:  Procedure LoadRessource( RessourceName : String; ImageList : TImageList)');
18064:  end;
18065:
18066:  procedure SIRegister_NMEA(CL: TPSPascalCompiler);
18067: begin
18068:  NMEADataArray', 'array of string');
18069:  Procedure TrimNMEA( var S : string)');
18070:  Procedure ExpandNMEA( var S : string)');
18071:  Function ParseNMEA( S : string) : NMEADataArray');
18072:  Function ChkValidNMEA( S : string) : Boolean');
18073:  Function IdNMEA( S : string) : string');
18074:  Function ChkSumNMEA( const S : string) : string');
18075:  Function PosInDeg( const PosStr : string) : Double');
18076:  Function DateTimeNMEA( const StrD, StrT : string) : TDateTime');
18077:  Function SysClockSet( const StrD, StrT : string) : Boolean');
18078:  function Ticks2Secs(Ticks : LongInt) : LongInt;');
18079:  function Secs2Ticks(Secs : LongInt) : LongInt;');
18080:  function MSecs2Ticks(MSecs : LongInt) : LongInt;');
18081: end;
18082:
18083: function TRestRequest_createStringStreamFromStringList(strings: TStringList): TStringStream;
18084:
18085: {A simple Oscilloscope using  TWaveIn class.
18086:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
18087: uses
18088:   Forms,
18089:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
18090:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
18091:   uColorFunctions in 'uColorFunctions.pas',
18092:   AMixer in 'AMixer.pas',
18093:   uSettings in 'uSettings.pas',
18094:   UWavein4 in 'UWavein4.pas',
18095:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
18096:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
18097:
18098: Functions_max hex in the box maXbox
18099: functionslist.txt
18100: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98/100/101
18101:
18102: ********************************************************************************
18103: Procedure
18104: PROCEDURE SIZE 8396 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
18105: Procedure ***************************Now the Procedure list*******************
18106: Procedure ( ACol, ARow : Integer; Items : TStrings)
18107: Procedure ( Agg : TAggregate)
18108: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
18109: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
18110: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
18111: Procedure ( ASender : TObject; const ABytes : Integer)
18112: Procedure ( ASender : TObject; VStream : TStream)
18113: Procedure ( AThread : TIdThread)
18114: Procedure ( AWebModule : TComponent)
18115: Procedure ( Column : TColumn)
18116: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
18117: Procedure ( const iStart : integer; const sText : string)
18118: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
18119: Procedure ( Database : TDatabase; LoginParams : TStrings)
18120: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
18121: Procedure ( DATASET : TDATASET)
18122: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
18123: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
18124: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
18125: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
18126: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
18127: Procedure ( Done : Integer)
18128: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
18129: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
18130: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
18131: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
18132: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
18133: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
18134: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
18135: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
18136: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
18137: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
18138: Procedure ( SENDER : TFIELD; const TEXT : String)
18139: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
18140: Procedure ( Sender : TIdTelnet; const Buffer : String)
18141: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
```

```
18142: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
18143: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18144: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
18145: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
18146: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
18147: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
18148: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
18149: Procedure ( Sender : TObject; Button : TMPBtnType)
18150: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
18151: Procedure ( Sender : TObject; Button : TUDBtnType)
18152: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
18153: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
18154: Procedure ( Sender : TObject; Column : TListColumn)
18155: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
18156: Procedure ( Sender : TObject; Connecting : Boolean)
18157: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
18158: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
18159: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
18160: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
18161: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
18162: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
18163: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
18164: Procedure ( Sender : TObject; Index : LongInt)
18165: Procedure ( Sender : TObject; Item : TListItem)
18166: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
18167: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
18168: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
18169: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
18170: Procedure ( Sender : TObject; Item : TListItem; var S : string)
18171: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
18172: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
18173: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
18174: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
18175: Procedure ( Sender : TObject; Node : TTreeNode)
18176: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
18177: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
18178: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
18179: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
18180: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
18181: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
18182: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
18183: Procedure ( Sender : TObject; Rect : TRect)
18184: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
18185: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
18186: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
18187: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
18188: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
18189: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
18190: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
18191: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
18192: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
18193: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
18194: Procedure ( Sender : TObject; Thread : TServerClientThread)
18195: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
18196: Procedure ( Sender : TObject; Username, Password : string)
18197: Procedure ( Sender : TObject; var AllowChange : Boolean)
18198: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
18199: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
18200: Procedure ( Sender : TObject; var Continue : Boolean)
18201: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
18202: Procedure ( Sender : TObject; var Username : string)
18203: Procedure ( Sender : TObject; Wnd : HWND)
18204: Procedure ( Sender : TToolbar; Button : TToolButton)
18205: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
18206: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
18207: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
18208: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
18209: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
18210: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
18211: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
18212: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
18213: procedure (Sender: TObject)
18214: procedure (Sender: TObject; var Done: Boolean)
18215: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
18216: procedure _T(Name: tbtString; v: Variant);
18217: Procedure AbandonSignalHandler( RtlSigNum : Integer)
18218: Procedure Abort
18219: Procedure About1Click( Sender : TObject)
18220: Procedure Accept( Socket : TSocket)
18221: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
18222: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
18223: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
18224: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
18225: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
18226: Procedure Add( Addend1, Addend2 : TMyBigInt)
18227: Procedure ADD( const AKEY, AVALUE : VARIANT)
18228: Procedure Add( const Key : string; Value : Integer)
18229: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
```

```
18230: Procedure ADD( FIELD : TFIELD)
18231: Procedure ADD( ITEM : TMENUITEM)
18232: Procedure ADD( POPUP : TPOPUPMENU)
18233: Procedure AddCharacters( xCharacters : TCharSet)
18234: Procedure AddDriver( const Name : string; List : TStrings)
18235: Procedure AddImages( Value : TCustomImageList)
18236: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
18237: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
18238: Procedure AddLoader( Loader : TBitmapLoader)
18239: Procedure ADDPARAM( VALUE : TPARAM)
18240: Procedure AddPassword( const Password : string)
18241: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
18242: Procedure AddState( oState : TniRegularExpressionState)
18243: Procedure AddStrings( Strings : TStrings);
18244: procedure AddStrings(Strings: TStrings);
18245: Procedure AddStrings1( Strings : TWideStrings);
18246: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
18247: Procedure AddToRecentDocs( const Filename : string)
18248: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
18249: Procedure AllFunctionsList1Click( Sender : TObject)
18250: procedure AllObjectsList1Click(Sender: TObject);
18251: Procedure Allocate( AAllocateBytes : Integer)
18252: procedure AllResourceList1Click(Sender: TObject);
18253: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
18254: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
18255: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
18256: Procedure AnsiFree( var s : AnsiString)
18257: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
18258: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
18259: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
18260: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
18261: Procedure AntiFreeze;
18262: Procedure APPEND
18263: Procedure Append( const S : WideString)
18264: procedure Append(S: string);
18265: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
18266: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
18267: Procedure AppendChunk( Val : OleVariant)
18268: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
18269: Procedure AppendStr( var Dest : string; S : string)
18270: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
18271: Procedure ApplyRange
18272: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18273: Procedure Arrange( Code : TListArrangement)
18274: procedure Assert(expr : Boolean; const msg: string);
18275: procedure Assert2(expr : Boolean; const msg: string);
18276: Procedure Assign( AList : TCustomBucketList)
18277: Procedure Assign( Other : TObject)
18278: Procedure Assign( Source : TDragObject)
18279: Procedure Assign( Source : TPersistent)
18280: Procedure Assign(Source: TPersistent)
18281: procedure Assign2(mystring, mypath: string);
18282: Procedure AssignCurValues( Source : TDataSet);
18283: Procedure AssignCurValues1( const CurValues : Variant);
18284: Procedure ASSIGNFIELD( FIELD : TFIELD)
18285: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
18286: Procedure AssignFile(var F: Text; FileName: string)
18287: procedure AssignFile(var F: TextFile; FileName: string)
18288: procedure AssignFileRead(var mystring, myfilename: string);
18289: procedure AssignFileWrite(mystring, myfilename: string);
18290: Procedure AssignTo( Other : TObject)
18291: Procedure AssignValues( Value : TParameters)
18292: Procedure ASSIGNVALUES( VALUE : TPARAMS)
18293: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
18294: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
18295: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
18296: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
18297: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18298: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
18299: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
18300: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
18301: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
18302: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
18303: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
18304: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
18305: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
18306: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
18307: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
18308: procedure Beep
18309: Procedure BeepOk
18310: Procedure BeepQuestion
18311: Procedure BeepHand
18312: Procedure BeepExclamation
18313: Procedure BeepAsterisk
18314: Procedure BeepInformation
18315: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
18316: Procedure BeginLayout
18317: Procedure BeginTimer( const Delay, Resolution : Cardinal)
18318: Procedure BeginUpdate
```

```
18319: procedure BeginUpdate;
18320: procedure BigScreen1Click(Sender: TObject);
18321: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
18322: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
18323: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
18324: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
18325: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
18326: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
18327: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
18328: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
18329: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
18330: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
18331: Procedure BreakPointMenuClick( Sender : TObject)
18332: procedure BRINGTOFRONT
18333: procedure BringToFront;
18334: Procedure btnBackClick( Sender : TObject)
18335: Procedure btnBrowseClick( Sender : TObject)
18336: Procedure BtnClick( Index : TNavigateBtn)
18337: Procedure btnLargeIconsClick( Sender : TObject)
18338: Procedure BuildAndSendRequest( AURI : TIdURI)
18339: Procedure BuildCache
18340: Procedure BurnMemory( var Buff, BuffLen : integer)
18341: Procedure BurnMemoryStream( Destructo : TMemoryStream)
18342: Procedure CalculateFirstSet
18343: Procedure Cancel
18344: procedure CancelDrag;
18345: Procedure CancelEdit
18346: procedure CANCELHINT
18347: Procedure CancelRange
18348: Procedure CancelUpdates
18349: Procedure CancelWriteBuffer
18350: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
18351: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
18352: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
18353: procedure CaptureScreenFormat(vname: string; vextension: string);
18354: procedure CaptureScreenPNG(vname: string);
18355: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
18356: procedure CASCADE
18357: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
18358: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
18359: Procedure cbPathClick( Sender : TObject)
18360: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18361: Procedure cedebugAfterExecute( Sender : TPSScript)
18362: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18363: Procedure cedebugCompile( Sender : TPSScript)
18364: Procedure cedebugExecute( Sender : TPSScript)
18365: Procedure cedebugIdle( Sender : TObject)
18366: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18367: Procedure CenterHeight( const pc, pcParent : TControl)
18368: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18369: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18370: Procedure Change
18371: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
18372: Procedure Changed
18373: Procedure ChangeDir( const ADirName : string)
18374: Procedure ChangeDirUp
18375: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
18376: Procedure ChangeLevelBy( Value : TChangeRange)
18377: Procedure ChDir(const s: string)
18378: Procedure Check(Status: Integer)
18379: Procedure CheckCommonControl( CC : Integer)
18380: Procedure CHECKFIELDNAME( const FIELDNAME : String)
18381: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
18382: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
18383: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
18384: Procedure CheckToken( T : Char)
18385: procedure CheckToken(t:char)
18386: Procedure CheckTokenSymbol( const S : string)
18387: procedure CheckTokenSymbol(s:string)
18388: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
18389: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18390: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
18391: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
18392: procedure CipherFile1Click(Sender: TObject);
18393: Procedure Clear;
18394: Procedure Clear1Click( Sender : TObject)
18395: Procedure ClearColor( Color : TColor)
18396: Procedure CLEARITEM( AITEM : TMENUITEM)
18397: Procedure ClearMapping
18398: Procedure ClearSelection( KeepPrimary : Boolean)
18399: Procedure ClearWriteBuffer
18400: Procedure Click
18401: Procedure Close
18402: Procedure Close1Click( Sender : TObject)
18403: Procedure CloseDatabase( Database : TDatabase)
18404: Procedure CloseDataSets
18405: Procedure CloseDialog
18406: Procedure CloseFile(var F: Text);
18407: Procedure Closure
```

```
18408: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18409: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18410: Procedure CodeCompletionList1Click( Sender : TObject)
18411: Procedure ColEnter
18412: Procedure Collapse
18413: Procedure Collapse( Recurse : Boolean)
18414: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
18415: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
18416: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
18417: Procedure Compile1Click( Sender : TObject)
18418: procedure ComponentCount1Click(Sender: TObject);
18419: Procedure Compress(azipfolder, azipfile: string)
18420: Procedure DeCompress(azipfolder, azipfile: string)
18421: Procedure XZip(azipfolder, azipfile: string)
18422: Procedure XUnZip(azipfolder, azipfile: string)
18423: Procedure Connect( const ATimeout : Integer)
18424: Procedure Connect( Socket : TSocket)
18425: procedure Console1Click(Sender: TObject);
18426: Procedure Continue
18427: Procedure ContinueCount( var Counter : TJclCounter)
18428: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
18429: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
18430: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
18431: Procedure ConvertImage(vsource, vdestination: string);
18432: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
18433: Procedure ConvertBitmap(vsource, vdestination: string);
18434: Procedure ConvertToGray(Cnv: TCanvas);
18435: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
18436: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
18437: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
18438: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
18439: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
18440: Procedure CopyFrom( mbCopy : TMyBigInt)
18441: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
18442: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
18443: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
18444: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
18445: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
18446: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
18447: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
18448: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
18449: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
18450: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18451: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
18452: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18453: Procedure CopyToClipboard
18454: Procedure CountParts
18455: Procedure CreateDataSet
18456: Procedure CreateEmptyFile( const FileName : string)
18457: Procedure CreateFileFromString( const FileName, Data : string)
18458: Procedure CreateFromDelta( Source : TPacketDataSet)
18459: procedure CREATEHANDLE
18460: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
18461: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
18462: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
18463: Procedure CreateTable
18464: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
18465: procedure CSyntax1Click(Sender: TObject);
18466: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
18467: Procedure CURSORPOSCHANGED
18468: procedure CutFirstDirectory(var S: String)
18469: Procedure DataBaseError(const Message: string)
18470: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
18471: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
18472: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
18473: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
18474: Procedure DBIError(errorCode: Integer)
18475: Procedure DebugOutput( const AText : string)
18476: Procedure DebugRun1Click( Sender : TObject)
18477: procedure Dec;
18478: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
18479: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
18480: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
18481: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
18482: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
18483: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
18484: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
18485: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
18486: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
18487: Procedure Decompile1Click( Sender : TObject)
18488: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
18489: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
18490: Procedure DeferLayout
18491: Procedure defFileread
18492: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
18493: Procedure DelayMicroseconds( const MicroSeconds : Integer)
```

```
18494: Procedure Delete
18495: Procedure Delete( const AFilename : string)
18496: Procedure Delete( const Index : Integer)
18497: Procedure DELETE( INDEX : INTEGER)
18498: Procedure Delete( Index : LongInt)
18499: Procedure Delete( Node : TTreeNode)
18500: procedure Delete(var s: AnyString; ifrom, icount: Longint);
18501: Procedure DeleteAlias( const Name : string)
18502: Procedure DeleteDriver( const Name : string)
18503: Procedure DeleteIndex( const Name : string)
18504: Procedure DeleteKey( const Section, Ident : String)
18505: Procedure DeleteRecords
18506: Procedure DeleteRecords( AffectRecords : TAffectRecords)
18507: Procedure DeleteString( var pStr : String; const pDelStr : string)
18508: Procedure DeleteTable
18509: procedure DelphiSite1Click(Sender: TObject);
18510: Procedure Deselect
18511: Procedure Deselect( Node : TTreeNode)
18512: procedure DestroyComponents
18513: Procedure DestroyHandle
18514: Procedure Diff( var X : array of Double)
18515: procedure Diff(var X: array of Double);
18516: Procedure DirCreate( const DirectoryName : String)');
18517: procedure DISABLEALIGN
18518: Procedure DisableConstraints
18519: Procedure Disconnect
18520: Procedure Disconnect( Socket : TSocket)
18521: Procedure Dispose
18522: procedure Dispose(P: PChar)
18523: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
18524: Procedure DoKey( Key : TDBCtrlGridKey)
18525: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18526: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18527: Procedure Dormant
18528: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
18529: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
18530: Procedure DoubleToComp( Value : Double; var Result : Comp)
18531: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18532: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
18533: procedure Draw(X, Y: Integer; Graphic: TGraphic);
18534: Procedure Draw1(Canvas:TCanvas; X,Y,
     Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
18535: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18536: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
18537: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18538: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
18539: procedure DrawFocusRect(const Rect: TRect);
18540: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
18541: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18542: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
18543: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
     TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
18544: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18545: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
18546: Procedure DropConnections
18547: Procedure DropDown
18548: Procedure DumpDescription( oStrings : TStrings)
18549: Procedure DumpStateTable( oStrings : TStrings)
18550: Procedure EDIT
18551: Procedure EditButtonClick
18552: Procedure EditFont1Click( Sender : TObject)
18553: procedure Ellipse(X1, Y1, X2, Y2: Integer);
18554: Procedure Ellipse1( const Rect : TRect);
18555: Procedure EMMS
18556: Procedure Encode( ADest : TStream)
18557: procedure ENDDRAG(DROP:BOOLEAN)
18558: Procedure EndEdit( Cancel : Boolean)
18559: Procedure EndTimer
18560: Procedure EndUpdate
18561: Procedure EraseSection( const Section : string)
18562: Procedure Error( const Ident : string)
18563: procedure Error(Ident:Integer)
18564: Procedure ErrorFmt( const Ident : string; const Args : array of const)
18565: Procedure ErrorStr( const Message : string)
18566: procedure ErrorStr(Message:String)
18567: Procedure Exchange( Index1, Index2 : Integer)
18568: procedure Exchange(Index1, Index2: Integer);
18569: Procedure Exec( FileName, Parameters, Directory : string)
18570: Procedure ExecProc
18571: Procedure ExecSQL( UpdateKind : TUpdateKind)
18572: Procedure Execute
18573: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
18574: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
18575: Procedure ExecuteCommand(executeFile, paramstring: string)
18576: Procedure ExecuteShell(executeFile, paramstring: string)
18577: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18578: Procedure ExitThread(ExitCode: Integer); stdcall;
18579: Procedure ExitProcess(ExitCode: Integer); stdcall;
18580: Procedure Expand( AUserName : String; AResults : TStrings)
```

```
18581: Procedure Expand( Recurse : Boolean)
18582: Procedure ExportClipboard1Click( Sender : TObject)
18583: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
18584: Procedure ExtractContentFields( Strings : TStrings)
18585: Procedure ExtractCookieFields( Strings : TStrings)
18586: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
18587: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
18588: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
18589: Procedure ExtractQueryFields( Strings : TStrings)
18590: Procedure FastDegToGrad
18591: Procedure FastDegToRad
18592: Procedure FastGradToDeg
18593: Procedure FastGradToRad
18594: Procedure FastRadToDeg
18595: Procedure FastRadToGrad
18596: Procedure FileClose( Handle : Integer)
18597: Procedure FileClose(handle: integer)
18598: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
18599: Procedure FileStructure( AStructure : TIdFTPDataStructure)
18600: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
18601: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
18602: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
18603: Procedure FillChar2(var X: PChar ; count: integer; value: char)
18604: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
18605: Procedure FillIPList
18606: procedure FillRect(const Rect: TRect);
18607: Procedure FillTStrings( AStrings : TStrings)
18608: Procedure FilterOnBookmarks( Bookmarks : array of const)
18609: procedure FinalizePackage(Module: HMODULE)
18610: procedure FindClose;
18611: procedure FindClose2(var F: TSearchRec)
18612: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
18613: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
18614: Procedure FindNearest( const KeyValues : array of const)
18615: Procedure FinishContext
18616: Procedure FIRST
18617: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
18618: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
18619: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
18620: Procedure FlushSchemaCache( const TableName : string)
18621: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
18622: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
18623: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
18624: Procedure FormActivate( Sender : TObject)
18625: procedure FormatLn(const format: String; const args: array of const);  //alias
18626: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18627: Procedure FormCreate( Sender : TObject)
18628: Procedure FormDestroy( Sender : TObject)
18629: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18630: procedure FormOutput1Click(Sender: TObject);
18631: Procedure FormToHtml( Form : TForm; Path : string)
18632: procedure FrameRect(const Rect: TRect);
18633: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
18634: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
18635: Procedure Free( Buffer : TRecordBuffer)
18636: Procedure Free( Buffer : TValueBuffer)
18637: Procedure Free;
18638: Procedure FreeAndNil(var Obj:TObject)
18639: Procedure FreeImage
18640: procedure FreeMem(P: PChar; Size: Integer)
18641: Procedure FreeTreeData( Tree : TUpdateTree)
18642: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
18643: Procedure FullCollapse
18644: Procedure FullExpand
18645: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
18646: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
18647: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
18648: Procedure Get1( AURL : string; const AResponseContent : TStream);
18649: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
18650: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
18651: Procedure GetAliasNames( List : TStrings)
18652: Procedure GetAliasParams( const AliasName : string; List : TStrings)
18653: Procedure GetApplicationsRunning( Strings : TStrings)
18654: Procedure getBox(aURL, extension: string);
18655: Procedure GetCommandTypes( List : TWideStrings)
18656: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
18657: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
18658: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
18659: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
18660: Procedure GetDatabaseNames( List : TStrings)
18661: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
18662: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
18663: Procedure GetDir(d: byte; var s: string)
18664: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
18665: Procedure GetDriverNames( List : TStrings)
18666: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
18667: Procedure GetDriverParams( const DriverName : string; List : TStrings)
```

```
18668: Procedure GetEMails1Click( Sender : TObject)
18669: Procedure getEnvironmentInfo;
18670: Function getEnvironmentString: string;
18671: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
18672: Procedure GetFieldNames( const TableName : string; List : TStrings)
18673: Procedure GetFieldNames( const TableName : string; List : TStrings);
18674: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
18675: Procedure GETFIELDNAMES( LIST : TSTRINGS)
18676: Procedure GetFieldNames1( const TableName : string; List : TStrings);
18677: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
18678: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
18679: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
18680: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
18681: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
18682: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
18683: Procedure GetFormatSettings
18684: Procedure GetFromDIB( var DIB : TBitmapInfo)
18685: Procedure GetFromHDIB( HDIB : HBitmap)
18686: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
18687: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
18688: Procedure GetIcon( Index : Integer; Image : TIcon);
18689: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
18690: Procedure GetIndexInfo( IndexName : string)
18691: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
18692: Procedure GetIndexNames( List : TStrings)
18693: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
18694: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
18695: Procedure GetIndexNames4( const TableName : string; List : TStrings);
18696: Procedure GetInternalResponse
18697: Procedure GETITEMNAMES( LIST : TSTRINGS)
18698: procedure GetMem(P: PChar; Size: Integer)
18699: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
18700: procedure GetPackageDescription(ModuleName: PChar): string)
18701: Procedure GetPackageNames( List : TStrings);
18702: Procedure GetPackageNames1( List : TWideStrings);
18703: Procedure GetParamList( List : TList; const ParamNames : WideString)
18704: Procedure GetProcedureNames( List : TStrings);
18705: Procedure GetProcedureNames( List : TWideStrings);
18706: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
18707: Procedure GetProcedureNames1( List : TStrings);
18708: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
18709: Procedure GetProcedureNames3( List : TWideStrings);
18710: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
18711: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
18712: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
18713: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
18714: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
18715: Procedure GetProviderNames( Names : TWideStrings);
18716: Procedure GetProviderNames( Proc : TGetStrProc)
18717: Procedure GetProviderNames1( Names : TStrings);
18718: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
18719: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
18720: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
18721: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
18722: Procedure GetSchemaNames( List : TStrings);
18723: Procedure GetSchemaNames1( List : TWideStrings);
18724: Procedure getScriptandRunAsk;
18725: Procedure getScriptandRun(ascript: string);
18726: Procedure getScript(ascript: string);   //alias
18727: Procedure getWebScript(ascript: string);   //alias
18728: Procedure GetSessionNames( List : TStrings)
18729: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
18730: Procedure GetStrings( List : TStrings)
18731: Procedure GetSystemTime; stdcall;
18732: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
18733: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
18734: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
18735: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
18736: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
18737: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
18738: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
18739: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
18740: Procedure GetVisibleWindows( List : Tstrings)
18741: Procedure GoBegin
18742: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
18743: Procedure GotoCurrent( Table : TTable)
18744: procedure GotoEnd1Click(Sender: TObject);
18745: Procedure GotoNearest
18746: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
18747: Procedure HandleException( E : Exception; var Handled : Boolean)
18748: procedure HANDLEMESSAGE
18749: procedure HandleNeeded;
18750: Procedure Head( AURL : string)
18751: Procedure Help( var AHelpContents : TStringList; ACommand : String)
18752: Procedure HexToBinary( Stream : TStream)
18753: procedure HexToBinary(Stream:TStream)
18754: Procedure HideDragImage
```

```
18755: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
18756: Procedure HideTraybar
18757: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
18758: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
18759: Procedure HookOSExceptions
18760: Procedure HookSignal( RtlSigNum : Integer)
18761: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
18762: Procedure HTMLSyntax1Click( Sender : TObject)
18763: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
18764: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
18765: Procedure ImportfromClipboard1Click( Sender : TObject)
18766: Procedure ImportfromClipboard2Click( Sender : TObject)
18767: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
18768: procedure Incb(var x: byte);
18769: Procedure Include1Click( Sender : TObject)
18770: Procedure IncludeOFF;  //preprocessing
18771: Procedure IncludeON;
18772: procedure Info1Click(Sender: TObject);
18773: Procedure InitAltRecBuffers( CheckModified : Boolean)
18774: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
18775: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
18776: Procedure InitData( ASource : TDataSet)
18777: Procedure InitDelta( ADelta : TPacketDataSet);
18778: Procedure InitDelta1( const ADelta : OleVariant);
18779: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
18780: Procedure Initialize
18781: procedure InitializePackage(Module: HMODULE)
18782: Procedure INITIATEACTION
18783: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
18784: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
18785: Procedure InitModule( AModule : TComponent)
18786: Procedure InitStdConvs
18787: Procedure InitTreeData( Tree : TUpdateTree)
18788: Procedure INSERT
18789: Procedure Insert( Index : Integer; AClass : TClass)
18790: Procedure Insert( Index : Integer; AComponent : TComponent)
18791: Procedure Insert( Index : Integer; AObject : TObject)
18792: Procedure Insert( Index : Integer; const S : WideString)
18793: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
18794: Procedure Insert(Index: Integer; const S: string);
18795: procedure Insert(Index: Integer; S: string);
18796: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
18797: procedure InsertComponent(AComponent:TComponent)
18798: procedure InsertControl(AControl: TControl);
18799: Procedure InsertIcon( Index : Integer; Image : TIcon)
18800: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
18801: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
18802: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
18803: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
18804: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
18805: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
18806: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
18807: Procedure InternalBeforeResolve( Tree : TUpdateTree)
18808: Procedure InvalidateModuleCache
18809: Procedure InvalidateTitles
18810: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
18811: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
18812: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
18813: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
18814: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
18815: procedure JavaSyntax1Click(Sender: TObject);
18816: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
18817: Procedure KillDataChannel
18818: Procedure Largefont1Click( Sender : TObject)
18819: Procedure LAST
18820: Procedure LaunchCpl( FileName : string)
18821: Procedure Launch( const AFile : string)
18822: Procedure LaunchFile( const AFile : string)
18823: Procedure LetFileList(FileList: TStringlist; apath: string);
18824: Procedure lineToNumber( xmemo : String; met : boolean)
18825: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
18826: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
18827: Procedure ListViewData( Sender : TObject; Item : TListItem)
18828: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
18829: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
18830: Procedure ListViewDblClick( Sender : TObject)
18831: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18832: Procedure ListDLLExports(const FileName: string; List: TStrings);
18833: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
18834: procedure LoadBytecode1Click(Sender: TObject);
18835: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
18836: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
18837: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
18838: Procedure LoadFromFile( AFileName : string)
18839: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
```

```
18840: Procedure LoadFromFile( const FileName : string)
18841: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
18842: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
18843: Procedure LoadFromFile( const FileName : WideString)
18844: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18845: Procedure LoadFromFile(const AFileName: string)
18846: procedure LoadFromFile(FileName: string);
18847: procedure LoadFromFile(FileName:String)
18848: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
18849: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
18850: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
18851: Procedure LoadFromStream( const Stream : TStream)
18852: Procedure LoadFromStream( S : TStream)
18853: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18854: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18855: Procedure LoadFromStream( Stream : TStream)
18856: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
18857: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
18858: procedure LoadFromStream(Stream: TStream);
18859: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
18860: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
18861: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
18862: Procedure LoadLastFile1Click( Sender : TObject)
18863: { LoadIcoToImage loads two icons from resource named NameRes,
18864:   into two image lists ALarge and ASmall}
18865: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
18866: Procedure LoadMemo
18867: Procedure LoadParamsFromIniFile( FFileName : WideString)
18868: Procedure Lock
18869: Procedure Login
18870: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
18871: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
18872: Procedure MakeCaseInsensitive
18873: Procedure MakeDeterministic( var bChanged : boolean)
18874: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
18875: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
18876: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
18877: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
18878: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
18879: Procedure SetRectComplexFormatStr( const S : string)
18880: Procedure SetPolarComplexFormatStr( const S : string)
18881: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
18882: Procedure MakeVisible
18883: Procedure MakeVisible( PartialOK : Boolean)
18884: Procedure Manual1Click( Sender : TObject)
18885: Procedure MarkReachable
18886: Procedure maXbox;  //shows the exe version data in a win box
18887: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
18888: Procedure Memo1Change( Sender : TObject)
18889: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
18890: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
18891: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18892: procedure Memory1Click(Sender: TObject);
18893: Procedure MERGE( MENU : TMAINMENU)
18894: Procedure MergeChangeLog
18895: procedure MINIMIZE
18896: Procedure MinimizeMaxbox;
18897: procedure MyCopyFile(Name1,Name2:string);
18898: Procedure MkDir(const s: string)
18899: Procedure MakeDir(const s: string)');
18900: Procedure ChangeDir(const s: string)');
18901: Function makeFile(const FileName: string): integer)');
18902: Procedure mnuPrintFont1Click( Sender : TObject)
18903: procedure ModalStarted
18904: Procedure Modified
18905: Procedure ModifyAlias( Name : string; List : TStrings)
18906: Procedure ModifyDriver( Name : string; List : TStrings)
18907: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
18908: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
18909: Procedure Move( CurIndex, NewIndex : Integer)
18910: procedure Move(CurIndex, NewIndex: Integer);
18911: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
18912: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
18913: Procedure moveCube( o : TMyLabel)
18914: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
18915: procedure MoveTo(X, Y: Integer);
18916: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
18917: Procedure MovePoint(var x,y:Extended; const angle:Extended);
18918: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
18919: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
18920: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
18921: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
18922: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
18923: procedure New(P: PChar)
18924: procedure New1Click(Sender: TObject);
18925: procedure NewInstance1Click(Sender: TObject);
18926: Procedure NEXT
18927: Procedure NextMonth
```

```
18928: Procedure Noop
18929: Procedure NormalizePath( var APath : string)
18930: procedure ObjectBinaryToText(Input, Output: TStream)
18931: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18932: procedure ObjectResourceToText(Input, Output: TStream)
18933: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18934: procedure ObjectTextToBinary(Input, Output: TStream)
18935: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18936: procedure ObjectTextToResource(Input, Output: TStream)
18937: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18938: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
18939: Procedure Open( const UserID : WideString; const Password : WideString);
18940: Procedure Open;
18941: Procedure open1Click( Sender : TObject)
18942: Procedure OpenCdDrive
18943: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
18944: Procedure OpenCurrent
18945: Procedure OpenFile(vfilenamepath: string)
18946: Procedure OpenDirectory1Click( Sender : TObject)
18947: Procedure OpenDir(adir: string);
18948: Procedure OpenIndexFile( const IndexName : string)
18949: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
18950: Procedure OpenWriteBuffer( const AThreshhold : Integer)
18951: Procedure OptimizeMem
18952: Procedure Options1( AURL : string);
18953: Procedure OutputDebugString(lpOutputString : PChar)
18954: Procedure PackBuffer
18955: Procedure Paint
18956: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
18957: Procedure PaintToTBitmap( Target : TBitmap)
18958: Procedure PaletteChanged
18959: Procedure ParentBiDiModeChanged
18960: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
18961: Procedure PasteFromClipboard;
18962: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
18963: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
18964: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
18965: Procedure PError( Text : string)
18966: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18967: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
18968: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
18969: procedure playmp3(mpath: string);
18970: Procedure PlayMP31Click( Sender : TObject)
18971: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
18972: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
18973: procedure PolyBezier(const Points: array of TPoint);
18974: procedure PolyBezierTo(const Points: array of TPoint);
18975: procedure Polygon(const Points: array of TPoint);
18976: procedure Polyline(const Points: array of TPoint);
18977: Procedure Pop
18978: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
18979: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
18980: Procedure POPUP( X, Y : INTEGER)
18981: Procedure PopupURL(URL : WideString);
18982: Procedure POST
18983: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
18984: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
18985: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
18986: Procedure PostUser( const Email, FirstName, LastName : WideString)
18987: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18988: procedure Pred(X: int64);
18989: Procedure Prepare
18990: Procedure PrepareStatement
18991: Procedure PreProcessXML( AList : TStrings)
18992: Procedure PreventDestruction
18993: Procedure Print( const Caption : string)
18994: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
18995: procedure printf(const format: String; const args: array of const);
18996: Procedure PrintList(Value: TStringList);
18997: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
18998: Procedure Printout1Click( Sender : TObject)
18999: Procedure ProcessHeaders
19000: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
19001: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
19002: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
19003: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
19004: Procedure ProcessMessagesOFF;  //application.processmessages
19005: Procedure ProcessMessagesON;
19006: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
19007: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
19008: Procedure Proclist Size is: 3797 /1415
19009: Procedure procMessClick( Sender : TObject)
19010: Procedure PSScriptCompile( Sender : TPSScript)
19011: Procedure PSScriptExecute( Sender : TPSScript)
19012: Procedure PSScriptLine( Sender : TObject)
19013: Procedure Push( ABoundary : string)
19014: procedure PushItem(AItem: Pointer)
19015: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
```

```
19016: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
19017: procedure PutLinuxLines(const Value: string)
19018: Procedure Quit
19019: Procedure RaiseConversionError( const AText : string);
19020: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
19021: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
19022: procedure RaiseException(Ex: TIFException; Param: String);
19023: Procedure RaiseExceptionForLastCmdResult;
19024: procedure RaiseLastException;
19025: procedure RaiseException2;
19026: Procedure RaiseException3(const Msg: string);
19027: Procedure RaiseExcept(const Msg: string);
19028: Procedure RaiseLastOSError
19029: Procedure RaiseLastWin32;
19030: procedure RaiseLastWin32Error)
19031: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
19032: Procedure RandomFillStream( Stream : TMemoryStream)
19033: procedure randomize;
19034: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
19035: Procedure RCS
19036: Procedure Read( Socket : TSocket)
19037: procedure Readln1(var ast: string);  //of inputquery
19038: Procedure ReadBlobData
19039: procedure ReadBuffer(Buffer:String;Count:LongInt)
19040: procedure ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
19041: Procedure ReadSection( const Section : string; Strings : TStrings)
19042: Procedure ReadSections( Strings : TStrings)
19043: Procedure ReadSections( Strings : TStrings);
19044: Procedure ReadSections1( const Section : string; Strings : TStrings);
19045: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
19046: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
19047: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
19048: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
19049: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
19050: Procedure Realign;
19051: procedure Rectangle(X1, Y1, X2, Y2: Integer);
19052: Procedure Rectangle1( const Rect : TRect);
19053: Procedure RectCopy( var Dest : TRect; const Source : TRect)
19054: Procedure RectFitToScreen( var R : TRect)
19055: Procedure RectGrow( var R : TRect; const Delta : Integer)
19056: Procedure RectGrowX( var R : TRect; const Delta : Integer)
19057: Procedure RectGrowY( var R : TRect; const Delta : Integer)
19058: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
19059: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
19060: Procedure RectNormalize( var R : TRect)
19061: //  TFileCallbackProcedure = procedure(filename:string);
19062: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
19063: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
19064: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
19065: Procedure Refresh;
19066: Procedure RefreshData( Options : TFetchOptions)
19067: Procedure REFRESHLOOKUPLIST
19068: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
19069: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
19070: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
19071: Procedure RegisterChanges( Value : TChangeLink)
19072: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
19073: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
19074: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
19075: Procedure ReInitialize( ADelay : Cardinal)
19076: procedure RELEASE
19077: Procedure Remove( const AByteCount : integer)
19078: Procedure REMOVE( FIELD : TFIELD)
19079: Procedure REMOVE( ITEM : TMENUITEM)
19080: Procedure REMOVE( POPUP : TPOPUPMENU)
19081: Procedure RemoveAllPasswords
19082: procedure RemoveComponent(AComponent:TComponent)
19083: Procedure RemoveDir( const ADirName : string)
19084: Procedure RemoveLambdaTransitions( var bChanged : boolean)
19085: Procedure REMOVEPARAM( VALUE : TPARAM)
19086: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
19087: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
19088: Procedure Rename( const ASourceFile, ADestFile : string)
19089: Procedure Rename( const FileName : string; Reload : Boolean)
19090: Procedure RenameTable( const NewTableName : string)
19091: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
19092: Procedure Replace1Click( Sender : TObject)
19093: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
19094: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
19095: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
19096: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
19097: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
19098: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
19099: Procedure Requery( Options : TExecuteOptions)
19100: Procedure Reset
19101: Procedure Reset1Click( Sender : TObject)
19102: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
19103: procedure ResourceExplore1Click(Sender: TObject);
19104: Procedure RestoreContents
```

```
19105: Procedure RestoreDefaults
19106: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
19107: Procedure RetrieveHeaders
19108: Procedure RevertRecord
19109: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
19110: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19111: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
19112: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
19113: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
19114: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
19115: Procedure RleCompress2( Stream : TStream)
19116: Procedure RleDecompress2( Stream : TStream)
19117: Procedure RmDir(const S: string)
19118: Procedure Rollback
19119: Procedure Rollback( TransDesc : TTransactionDesc)
19120: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
19121: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
19122: Procedure RollbackTrans
19123: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
19124: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
19125: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
19126: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
19127: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
19128: Procedure S_EBox( const AText : string)
19129: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
19130: Procedure S_IBox( const AText : string)
19131: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
19132: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
19133: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
19134: Procedure SampleVarianceAndMean
19135: ( const X : TDynFloatArray; var Variance, Mean : Float)
19136: Procedure Save2Click( Sender : TObject)
19137: Procedure Saveas3Click( Sender : TObject)
19138: Procedure Savebefore1Click( Sender : TObject)
19139: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
19140: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19141: Procedure SaveConfigFile
19142: Procedure SaveOutput1Click( Sender : TObject)
19143: procedure SaveScreenshotClick(Sender: TObject);
19144: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
19145: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
19146: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
19147: Procedure SaveToFile( AFileName : string)
19148: Procedure SAVETOFILE( const FILENAME : String)
19149: Procedure SaveToFile( const FileName : WideString)
19150: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
19151: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
19152: procedure SaveToFile(FileName: string);
19153: procedure SaveToFile(FileName:String)
19154: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
19155: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
19156: Procedure SaveToStream( S : TStream)
19157: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
19158: Procedure SaveToStream( Stream : TStream)
19159: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
19160: procedure SaveToStream(Stream: TStream);
19161: procedure SaveToStream(Stream:TStream)
19162: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
19163: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
19164: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
19165: procedure Say(const sText: string)
19166: Procedure SBytecode1Click( Sender : TObject)
19167: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
19168: procedure ScriptExplorer1Click(Sender: TObject);
19169: Procedure Scroll( Distance : Integer)
19170: Procedure Scroll( DX, DY : Integer)
19171: procedure ScrollBy(DeltaX, DeltaY: Integer);
19172: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
19173: Procedure ScrollTabs( Delta : Integer)
19174: Procedure Search1Click( Sender : TObject)
19175: procedure SearchAndOpenDoc(vfilenamepath: string)
19176: procedure SearchAndOpenFile(vfilenamepath: string)
19177: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
19178: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19179: Procedure SearchNext1Click( Sender : TObject)
19180: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
19181: Procedure Select1( const Nodes : array of TTreeNode);
19182: Procedure Select2( Nodes : TList);
19183: Procedure SelectNext( Direction : Boolean)
19184: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
19185: Procedure SelfTestPEM  //unit uPSI_cPEM
19186: Procedure Send( AMsg : TIdMessage)
19187: //config forst in const MAILINIFILE = 'maildef.ini';
19188: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
19189: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19190: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19191: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
19192: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
```

```
19193: Procedure SendResponse
19194: Procedure SendStream( AStream : TStream)
19195: Procedure Set8087CW( NewCW : Word)
19196: Procedure SetAll( One, Two, Three, Four : Byte)
19197: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
19198: Procedure SetAppDispatcher( const ADispatcher : TComponent)
19199: procedure SetArrayLength;
19200: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
19201: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19202: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer);  //all init
19203: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
19204: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);');
19205: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);');
19206: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
19207: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
19208: Procedure SetAsHandle( Format : Word; Value : THandle)
19209: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
19210: procedure SetCaptureControl(Control: TControl);
19211: Procedure SetColumnAttributes
19212: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
19213: Procedure SetCustomHeader( const Name, Value : string)
19214: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
19215: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
19216: Procedure SetFocus
19217: procedure SetFocus; virtual;
19218: Procedure SetInitialState
19219: Procedure SetKey
19220: procedure SetLastError(ErrorCode: Integer)
19221: procedure SetLength;
19222: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
19223: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
19224: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
19225: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
19226: Procedure SetParams1( UpdateKind : TUpdateKind);
19227: Procedure SetPassword( const Password : string)
19228: Procedure SetPointer( Ptr : Pointer; Size : Longint)
19229: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
19230: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
19231: Procedure SetProvider( Provider : TComponent)
19232: Procedure SetProxy( const Proxy : string)
19233: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
19234: Procedure SetRange( const StartValues, EndValues : array of const)
19235: Procedure SetRangeEnd
19236: Procedure SetRate( const aPercent, aYear : integer)
19237: procedure SetRate(const aPercent, aYear: integer)
19238: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19239: Procedure SetSafeCallExceptionMsg( Msg : String)
19240: procedure SETSELTEXTBUF(BUFFER:PCHAR)
19241: Procedure SetSize( AWidth, AHeight : Integer)
19242: procedure SetSize(NewSize:LongInt)
19243: procedure SetString(var s: string; buffer: PChar; len: Integer)
19244: Procedure SetStrings( List : TStrings)
19245: Procedure SetText( Text : PwideChar)
19246: procedure SetText(Text: PChar);
19247: Procedure SetTextBuf( Buffer : PChar)
19248: procedure SETTEXTBUF(BUFFER:PCHAR)
19249: Procedure SetTick( Value : Integer)
19250: Procedure SetTimeout( ATimeOut : Integer)
19251: Procedure SetTraceEvent( Event : TDBXTraceEvent)
19252: Procedure SetUserName( const UserName : string)
19253: Procedure SetWallpaper( Path : string);
19254: procedure ShellStyle1Click(Sender: TObject);
19255: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
19256: Procedure ShowFileProperties( const FileName : string)
19257: Procedure ShowInclude1Click( Sender : TObject)
19258: Procedure ShowInterfaces1Click( Sender : TObject)
19259: Procedure ShowLastException1Click( Sender : TObject)
19260: Procedure ShowMessage( const Msg : string)
19261: Procedure ShowMessageBig(const aText : string);
19262: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
19263: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
19264: Procedure MsgBig(const aText : string);        //alias
19265: procedure showmessage(mytext: string);
19266: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
19267: procedure ShowMessageFmt(const Msg: string; Params: array of const))
19268: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
19269: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
19270: Procedure ShowSearchDialog( const Directory : string)
19271: Procedure ShowSpecChars1Click( Sender : TObject)
19272: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
19273: Procedure ShredFile( const FileName : string; Times : Integer)
19274: procedure Shuffle(vQ: TStringList);
19275: Procedure ShuffleList( var List : array of Integer; Count : Integer)
19276: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
19277: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
19278: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
19279: Procedure Site( const ACommand : string)
19280: Procedure SkipEOL
```

```
19281: Procedure Sleep( ATime : cardinal)
19282: Procedure Sleep( milliseconds : Cardinal)
19283: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
19284: Procedure Slinenumbers1Click( Sender : TObject)
19285: Procedure Sort
19286: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
19287: procedure SoundAlarm;  //beep seq
19288: procedure Speak(const sText: string)   //async like voice
19289: procedure Speak2(const sText: string)  //sync
19290: procedure Split(Str: string;  SubStr: string; List: TStrings);
19291: Procedure SplitNameValue( const Line : string; var Name, Value : string)
19292: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
19293: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
19294: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
19295: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
19296: procedure SQLSyntax1Click(Sender: TObject);
19297: Procedure SRand( Seed : RNG_IntType)
19298: Procedure Start
19299: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
19300: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
19301: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
19302: Procedure StartTransaction( TransDesc : TTransactionDesc)
19303: Procedure Status( var AStatusList : TStringList)
19304: Procedure StatusBar1DblClick( Sender : TObject)
19305: Procedure StepInto1Click( Sender : TObject)
19306: Procedure StepIt
19307: Procedure StepOut1Click( Sender : TObject)
19308: Procedure Stop
19309: procedure stopmp3;
19310: procedure StartWeb(aurl: string);
19311: Procedure Str(aint: integer; astr: string); //of system
19312: Procedure StrDispose( Str : PChar)
19313: procedure StrDispose(Str: PChar)
19314: Procedure StrReplace(var Str: String; Old, New: String);
19315: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
19316: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
19317: Procedure StringToBytes( Value : String; Bytes : array of byte)
19318: procedure StrSet(c : Char; I : Integer; var s : String);
19319: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19320: Procedure StructureMount( APath : String)
19321: procedure STYLECHANGED(SENDER:TOBJECT)
19322: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
19323: procedure Succ(X: int64);
19324: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
19325: procedure SwapChar(var X,Y: char);  //swapX follows
19326: Procedure SwapFloats( var X, Y : Float)
19327: procedure SwapGrid(grd: TStringGrid);
19328: Procedure SwapOrd( var I, J : Byte);
19329: Procedure SwapOrd( var X, Y : Integer)
19330: Procedure SwapOrd1( var I, J : Shortint);
19331: Procedure SwapOrd2( var I, J : Smallint);
19332: Procedure SwapOrd3( var I, J : Word);
19333: Procedure SwapOrd4( var I, J : Integer);
19334: Procedure SwapOrd5( var I, J : Cardinal);
19335: Procedure SwapOrd6( var I, J : Int64);
19336: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
19337: Procedure Synchronize1( Method : TMethod);
19338: procedure SyntaxCheck1Click(Sender: TObject);
19339: Procedure SysFreeString(const S: WideString); stdcall;
19340: Procedure TakeOver( Other : TLinearBitmap)
19341: Procedure Talkln(const sText: string)  //async voice
19342: procedure tbtn6resClick(Sender: TObject);
19343: Procedure tbtnUseCaseClick( Sender : TObject)
19344: procedure TerminalStyle1Click(Sender: TObject);
19345: Procedure Terminate
19346: Procedure texSyntax1Click( Sender : TObject)
19347: procedure TextOut(X, Y: Integer; Text: string);
19348: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
19349: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
19350: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
19351: Procedure TextStart
19352: procedure TILE
19353: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
19354: Procedure TitleClick( Column : TColumn)
19355: Procedure ToDo
19356: procedure toolbtnTutorialClick(Sender: TObject);
19357: Procedure Trace1( AURL : string; const AResponseContent : TStream);
19358: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
19359: Procedure Truncate
19360: procedure Tutorial101Click(Sender: TObject);
19361: procedure Tutorial10Statistics1Click(Sender: TObject);
19362: procedure Tutorial11Forms1Click(Sender: TObject);
19363: procedure Tutorial12SQL1Click(Sender: TObject);
19364: Procedure tutorial1Click( Sender : TObject)
19365: Procedure tutorial21Click( Sender : TObject)
19366: Procedure tutorial31Click( Sender : TObject)
19367: Procedure tutorial4Click( Sender : TObject)
19368: Procedure Tutorial5Click( Sender : TObject)
19369: procedure Tutorial6Click(Sender: TObject);
```

```
19370: procedure Tutorial91Click(Sender: TObject);
19371: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
19372: procedure UniqueString(var str: AnsiString)
19373: procedure UnloadLoadPackage(Module: HMODULE)
19374: Procedure Unlock
19375: Procedure UNMERGE( MENU : TMAINMENU)
19376: Procedure UnRegisterChanges( Value : TChangeLink)
19377: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
19378: Procedure UnregisterConversionType( const AType : TConvType)
19379: Procedure UnRegisterProvider( Prov : TCustomProvider)
19380: Procedure UPDATE
19381: Procedure UpdateBatch( AffectRecords : TAffectRecords)
19382: Procedure UPDATECURSORPOS
19383: Procedure UpdateFile
19384: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
19385: Procedure UpdateResponse( AResponse : TWebResponse)
19386: Procedure UpdateScrollBar
19387: Procedure UpdateView1Click( Sender : TObject)
19388: procedure Val(const s: string; var n, z: Integer)
19389: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
19390: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
19391: Procedure VariantAdd( const src : Variant; var dst : Variant)
19392: Procedure VariantAnd( const src : Variant; var dst : Variant)
19393: Procedure VariantArrayRedim( var V : Variant; High : Integer)
19394: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
19395: Procedure VariantClear( var V : Variant)
19396: Procedure VariantCpy( const src : Variant; var dst : Variant)
19397: Procedure VariantDiv( const src : Variant; var dst : Variant)
19398: Procedure VariantMod( const src : Variant; var dst : Variant)
19399: Procedure VariantMul( const src : Variant; var dst : Variant)
19400: Procedure VariantOr( const src : Variant; var dst : Variant)
19401: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
19402: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
19403: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
19404: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
19405: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
19406: Procedure VariantShl( const src : Variant; var dst : Variant)
19407: Procedure VariantShr( const src : Variant; var dst : Variant)
19408: Procedure VariantSub( const src : Variant; var dst : Variant)
19409: Procedure VariantXor( const src : Variant; var dst : Variant)
19410: Procedure VarCastError;
19411: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
19412: Procedure VarInvalidOp
19413: Procedure VarInvalidNullOp
19414: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
19415: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
19416: Procedure VarArrayCreateError
19417: Procedure VarResultCheck( AResult : HRESULT);
19418: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
19419: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
19420: Function VarTypeAsText( const AType : TVarType) : string
19421: procedure Voice(const sText: string) //async
19422: procedure Voice2(const sText: string) //sync
19423: Procedure WaitMiliSeconds( AMSec : word)
19424: Procedure WaitMS( AMSec : word)');
19425: procedure WebCamPic(picname: string);  //eg: c:\mypic.png
19426: Procedure WideAppend( var dst : WideString; const src : WideString)
19427: Procedure WideAssign( var dst : WideString; var src : WideString)
19428: Procedure WideDelete( var dst : WideString; index, count : Integer)
19429: Procedure WideFree( var s : WideString)
19430: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
19431: Procedure WideFromPChar( var dst : WideString; src : PChar)
19432: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
19433: Procedure WideSetLength( var dst : WideString; len : Integer)
19434: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
19435: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
19436: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
19437: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19438: Procedure HttpGet(const Url: string; Stream:TStream);
19439: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
19440: Procedure WordWrap1Click( Sender : TObject)
19441: Procedure Write( const AOut : string)
19442: Procedure Write( Socket : TSocket)
19443: procedure Write(S: string);
19444: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
19445: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
19446: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
19447: procedure WriteBuffer(Buffer:String;Count:LongInt)
19448: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
19449: Procedure WriteChar( AValue : Char)
19450: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
19451: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
19452: Procedure WriteFloat( const Section, Name : string; Value : Double)
19453: Procedure WriteHeader( AHeader : TStrings)
19454: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
19455: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
19456: Procedure WriteLn( const AOut : string)
19457: procedure Writeln(s: string);
19458: Procedure WriteLog( const FileName, LogLine : string)
```

```
19459: Procedure WriteRFCReply( AReply : TIdRFCReply)
19460: Procedure WriteRFCStrings( AStrings : TStrings)
19461: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
19462: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
19463: Procedure WriteString( const Section, Ident, Value : String)
19464: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
19465: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
19466: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
19467: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19468: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19469: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
19470: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
19471: procedure XMLSyntax1Click(Sender: TObject);
19472: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
19473: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
19474: Procedure ZeroFillStream( Stream : TMemoryStream)
19475: procedure XMLSyntax1Click(Sender: TObject);
19476: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
19477: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
19478: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
19479: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
19480: procedure(Sender, Source: TObject;X, Y: Integer)
19481: procedure(Sender, Target: TObject; X, Y: Integer)
19482: procedure(Sender: TObject; ASection, AWidth: Integer)
19483: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
19484: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
19485: procedure(Sender: TObject; var Action: TCloseAction)
19486: procedure(Sender: TObject; var CanClose: Boolean)
19487: procedure(Sender: TObject; var Key: Char);
19488: ProcedureName ProcedureNames ProcedureParametersCursor @
19489:
19490: *************Now Constructors constructor *************
19491:  Size is: 1248 1115 996 628 550 544 501 459 (381)
19492:  Attach( VersionInfoData : Pointer; Size : Integer)
19493:  constructor Create( ABuckets : TBucketListSizes)
19494:  Create( ACallBackWnd : HWND)
19495:  Create( AClient : TCustomTaskDialog)
19496:  Create( AClient : TIdTelnet)
19497:  Create( ACollection : TCollection)
19498:  Create( ACollection : TFavoriteLinkItems)
19499:  Create( ACollection : TTaskDialogButtons)
19500:  Create( AConnection : TIdCustomHTTP)
19501:  Create( ACreateSuspended : Boolean)
19502:  Create( ADataSet : TCustomSQLDataSet)
19503:  CREATE( ADATASET : TDATASET)
19504:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19505:  Create( AGrid : TCustomDBGrid)
19506:  Create( AGrid : TStringGrid; AIndex : Longint)
19507:  Create( AHTTP : TIdCustomHTTP)
19508:  Create( AListItems : TListItems)
19509:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19510:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19511:  Create( AOwner : TCommonCalendar)
19512:  Create( AOwner : TComponent)
19513:  CREATE( AOWNER : TCOMPONENT)
19514:  Create( AOwner : TCustomListView)
19515:  Create( AOwner : TCustomOutline)
19516:  Create( AOwner : TCustomRichEdit)
19517:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
19518:  Create( AOwner : TCustomTreeView)
19519:  Create( AOwner : TIdUserManager)
19520:  Create( AOwner : TListItems)
19521:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19522:  CREATE( AOWNER : TPERSISTENT)
19523:  Create( AOwner : TPersistent)
19524:  Create( AOwner : TTable)
19525:  Create( AOwner : TTreeNodes)
19526:  Create( AOwner : TWinControl; const ClassName : string)
19527:  Create( AParent : TIdCustomHTTP)
19528:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
19529:  Create( AProvider : TBaseProvider)
19530:  Create( AProvider : TCustomProvider);
19531:  Create( AProvider : TDataSetProvider)
19532:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
19533:  Create( ASocket : TSocket)
19534:  Create( AStrings : TWideStrings)
19535:  Create( AToolBar : TToolBar)
19536:  Create( ATreeNodes : TTreeNodes)
19537:  Create( Autofill : boolean)
19538:  Create( AWebPageInfo : TAbstractWebPageInfo)
19539:  Create( AWebRequest : TWebRequest)
19540:  Create( Collection : TCollection)
19541:  Create( Collection : TIdMessageParts; ABody : TStrings)
19542:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
19543:  Create( Column : TColumn)
19544:  Create( const AConvFamily : TConvFamily; const ADescription : string)
19545:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
19546:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
```

```
19547:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
19548:  Create( const ATabSet : TTabSet)
19549:  Create( const Compensate : Boolean)
19550:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
19551:  Create( const FileName : string)
19552:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
19553:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
19554:  Create( const MaskValue : string)
19555:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
19556:  Create( const Prefix : string)
19557:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
19558:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19559:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19560:  Create( CoolBar : TCoolBar)
19561:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
19562:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
19563:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
19564:  Create( DBCtrlGrid : TDBCtrlGrid)
19565:  Create( DSTableProducer : TDSTableProducer)
19566:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
19567:  Create( ErrorCode : DBIResult)
19568:  Create( Field : TBlobField; Mode : TBlobStreamMode)
19569:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
19570:  Create( HeaderControl : TCustomHeaderControl)
19571:  Create( HTTPRequest : TWebRequest)
19572:  Create( iStart : integer; sText : string)
19573:  Create( iValue : Integer)
19574:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
19575:  Create( MciErrNo : MCIERROR; const Msg : string)
19576:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19577:  Create( Message : string; ErrorCode : DBResult)
19578:  Create( Msg : string)
19579:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
19580:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
19581:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
19582:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
19583:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
19584:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
19585:  Create( Owner : TCustomComboBoxEx)
19586:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
19587:  Create( Owner : TPersistent)
19588:  Create( Params : TStrings)
19589:  Create( Size : Cardinal)
19590:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
19591:  Create( StatusBar : TCustomStatusBar)
19592:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
19593:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
19594:  Create(AHandle:Integer)
19595:  Create(AOwner: TComponent); virtual;
19596:  Create(const AURI : string)
19597:  Create(FileName:String;Mode:Word)
19598:  Create(Instance:THandle;ResName:String;ResType:PChar)
19599:  Create(Stream : TStream)
19600:  Create1( ADataset : TDataset);
19601:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
19602:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19603:  Create2( Other : TObject);
19604:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19605:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19606:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
19607:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
19608:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19609:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
19610:  CreateRes( Ident : Integer);
19611:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
19612:  CreateRes( ResStringRec : PResStringRec);
19613:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
19614:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19615:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19616:  CreateSize( AWidth, AHeight : Integer)
19617:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
19618:
19619: ------------------------------------------------------------------------------
19620: unit uPSI_MathMax;
19621: ------------------------------------------------------------------------------
19622:  CONSTS
19623:   Bernstein: Float = 0.28016949902386913303643649123 07;  // Bernstein constant
19624:   Cbrt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
19625:   Cbrt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
19626:   Cbrt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
19627:   Cbrt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
19628:   CbrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
19629:   Catalan: Float   = 0.91596559417721901505460351493 24;  // Catalan constant
19630:   PiJ:   Float       = 3.14159265358979323846264338327 95;  // PI
19631:   PI:  Extended =  3.14159265358979323846264338327 95);
```

```
19632:   PiOn2: Float     = 1.57079632679489661923132169163975;  // PI / 2
19633:   PiOn3: Float     = 1.04719755119659774615421446109317;  // PI / 3
19634:   PiOn4: Float     = 0.78539816339744830961566084581988; // PI / 4
19635:   Sqrt2: Float     = 1.41421356237309504880168872420969;  // Sqrt(2)
19636:   Sqrt3: Float     = 1.73205080756887729352744634150587;  // Sqrt(3)
19637:   Sqrt5: Float     = 2.23606797749978969640917366873127;  // Sqrt(5)
19638:   Sqrt10: Float    = 3.16227766016837933199889354443271;  // Sqrt(10)
19639:   SqrtPi: Float    = 1.77245385090551602729816748334115;  // Sqrt(PI)
19640:   Sqrt2Pi: Float   = 2.50662827463100050241576528481104;   // Sqrt(2 * PI)
19641:   TwoPi: Float     = 6.28318530717958647692528676655901;   // 2 * PI
19642:   ThreePi: Float   = 9.42477796076937971538793014983835;  // 3 * PI
19643:   Ln2: Float       = 0.69314718055994530941723212145818; // Ln(2)
19644:   Ln10: Float      = 2.30258509299404568401799145468436;  // Ln(10)
19645:   LnPi: Float      = 1.14472988584940017414342735135305;  // Ln(PI)
19646:   Log2J: Float     = 0.30102999566398119521373889472449; // Log10(2)
19647:   Log3: Float      = 0.47712125471966243729502790325512; // Log10(3)
19648:   LogPi: Float     = 0.49714987269413385435126828829091;  // Log10(PI)
19649:   LogE: Float      = 0.43429448190325182765112891891661; // Log10(E)
19650:   E: Float         = 2.71828182845904523536028747135266;  // Natural constant
19651:   hLn2Pi: Float    = 0.91893853320467274178032973640562; // Ln(2*PI)/2
19652:   inv2Pi: Float    = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
19653:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
19654:   GoldenMean: Float   = 1.61803398874989484820458683436563811;  // GoldenMean
19655:   EulerMascheroni: Float = 0.57721566490153286060651209008240243;  // Euler GAMMA
19656:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
19657:   StDelta         : Extended = 0.00001;  {delta for difference equations}
19658:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
19659:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
19660:
19661: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
19662: begin
19663:   MetersPerInch = 0.0254; // [1]
19664:   MetersPerFoot = MetersPerInch * 12;
19665:   MetersPerYard = MetersPerFoot * 3;
19666:   MetersPerMile = MetersPerFoot * 5280;
19667:   MetersPerNauticalMiles = 1852;
19668:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
19669:   MetersPerLightSecond = 2.99792458E8; // [5]
19670:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
19671:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
19672:   MetersPerCubit = 0.4572; // [6][7]
19673:   MetersPerFathom = MetersPerFoot * 6;
19674:   MetersPerFurlong = MetersPerYard * 220;
19675:   MetersPerHand = MetersPerInch * 4;
19676:   MetersPerPace = MetersPerInch * 30;
19677:   MetersPerRod = MetersPerFoot * 16.5;
19678:   MetersPerChain = MetersPerRod * 4;
19679:   MetersPerLink = MetersPerChain / 100;
19680:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
19681:   MetersPerPica = MetersPerPoint * 12;
19682:
19683:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
19684:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
19685:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
19686:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
19687:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
19688:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
19689:
19690:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
19691:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
19692:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
19693:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
19694:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
19695:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
19696:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
19697:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
19698:
19699:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
19700:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
19701:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
19702:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
19703:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
19704:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
19705:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
19706:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
19707:
19708:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
19709:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
19710:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
19711:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
19712:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
19713:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
19714:
19715:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
19716:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
19717:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
19718:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
19719:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
19720:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
```

```
19721:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
19722:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
19723:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
19724:
19725:   GramsPerPound = 453.59237; // [1][7]
19726:   GramsPerDrams = GramsPerPound / 256;
19727:   GramsPerGrains = GramsPerPound / 7000;
19728:   GramsPerTons = GramsPerPound * 2000;
19729:   GramsPerLongTons = GramsPerPound * 2240;
19730:   GramsPerOunces = GramsPerPound / 16;
19731:   GramsPerStones = GramsPerPound * 14;
19732:
19733:   MaxAngle 9223372036854775808.0;
19734:   MaxTanH 5678.26170314707197474596553898 54);
19735:   MaxFactorial( 1754);
19736:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
19737:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
19738:   MaxTanH( 354.89135644669199842162284618659);
19739:   MaxFactorial'LongInt'( 170);
19740:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
19741:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
19742:   MaxTanH( 44.361419555836499802702855773323);
19743:   MaxFactorial'LongInt'( 33);
19744:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
19745:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
19746:   PiExt( 3.14159265358979323846264338327 95);
19747:   RatioDegToRad( PiExt / 180.0);
19748:   RatioGradToRad( PiExt / 200.0);
19749:   RatioDegToGrad( 200.0 / 180.0);
19750:   RatioGradToDeg( 180.0 / 200.0);
19751:   Crc16PolynomCCITT'LongWord $1021);
19752:  Crc16PolynomIBM'LongWord $8005);
19753:  Crc16Bits'LongInt'( 16);
19754:  Crc16Bytes'LongInt'( 2);
19755:  Crc16HighBit'LongWord $8000);
19756:  NotCrc16HighBit','LongWord $7FFF);
19757:   Crc32PolynomIEEE','LongWord $04C11DB7);
19758:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
19759:  Crc32Koopman','LongWord $741B8CD7);
19760:  Crc32Bits','LongInt'( 32);
19761:  Crc32Bytes','LongInt'( 4);
19762:  Crc32HighBit','LongWord $80000000);
19763:  NotCrc32HighBit','LongWord $7FFFFFFF);
19764:
19765:   MinByte       = Low(Byte);
19766:   MaxByte       = High(Byte);
19767:   MinWord       = Low(Word);
19768:   MaxWord       = High(Word);
19769:   MinShortInt   = Low(ShortInt);
19770:   MaxShortInt   = High(ShortInt);
19771:   MinSmallInt   = Low(SmallInt);
19772:   MaxSmallInt   = High(SmallInt);
19773:   MinLongWord   = LongWord(Low(LongWord));
19774:   MaxLongWord   = LongWord(High(LongWord));
19775:   MinLongInt    = LongInt(Low(LongInt));
19776:   MaxLongInt    = LongInt(High(LongInt));
19777:   MinInt64      = Int64(Low(Int64));
19778:   MaxInt64      = Int64(High(Int64));
19779:   MinInteger    = Integer(Low(Integer));
19780:   MaxInteger    = Integer(High(Integer));
19781:   MinCardinal   = Cardinal(Low(Cardinal));
19782:   MaxCardinal   = Cardinal(High(Cardinal));
19783:   MinNativeUInt = NativeUInt(Low(NativeUInt));
19784:   MaxNativeUInt = NativeUInt(High(NativeUInt));
19785:   MinNativeInt  = NativeInt(Low(NativeInt));
19786:   MaxNativeInt  = NativeInt(High(NativeInt));
19787:  Function CosH( const Z : Float) : Float;
19788:  Function SinH( const Z : Float) : Float;
19789:  Function TanH( const Z : Float) : Float;
19790:
19791:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
19792:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
19793:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
19794:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
19795:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
19796:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
19797:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
19798:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
19799:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
19800:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
19801:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
19802:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
19803:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
19804:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
19805:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
19806:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
19807:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
19808:   MaxLog = 709.7827128933840;
19809:   MinLog = -708.3964185322641;
```

```
19810:   MaxFac = 170;
19811:   MaxGam = 171.624376956302;
19812:   MaxLgm = 2.556348E+305;
19813:   SingleCompareDelta   = 1.0E-34;
19814:   DoubleCompareDelta   = 1.0E-280;
19815:   {$IFDEF CLR}
19816:   ExtendedCompareDelta = DoubleCompareDelta;
19817:   {$ELSE}
19818:   ExtendedCompareDelta = 1.0E-4400;
19819:   {$ENDIF}
19820:   Bytes1KB  = 1024;
19821:   Bytes1MB  = 1024 * Bytes1KB;
19822:   Bytes1GB  = 1024 * Bytes1MB;
19823:   Bytes64KB = 64 * Bytes1KB;
19824:   Bytes64MB = 64 * Bytes1MB;
19825:   Bytes2GB  = 2 * LongWord(Bytes1GB);
19826:     clBlack32', $FF000000 ));
19827:     clDimGray32', $FF3F3F3F ));
19828:     clGray32', $FF7F7F7F ));
19829:     clLightGray32', $FFBFBFBF ));
19830:     clWhite32', $FFFFFFFF ));
19831:     clMaroon32', $FF7F0000 ));
19832:     clGreen32', $FF007F00 ));
19833:     clOlive32', $FF7F7F00 ));
19834:     clNavy32', $FF00007F ));
19835:     clPurple32', $FF7F007F ));
19836:     clTeal32', $FF007F7F ));
19837:     clRed32', $FFFF0000 ));
19838:     clLime32', $FF00FF00 ));
19839:     clYellow32', $FFFFFF00 ));
19840:     clBlue32', $FF0000FF ));
19841:     clFuchsia32', $FFFF00FF ));
19842:     clAqua32', $FF00FFFF ));
19843:     clAliceBlue32', $FFF0F8FF ));
19844:     clAntiqueWhite32', $FFFAEBD7 ));
19845:     clAquamarine32', $FF7FFFD4 ));
19846:     clAzure32', $FFF0FFFF ));
19847:     clBeige32', $FFF5F5DC ));
19848:     clBisque32', $FFFFE4C4 ));
19849:     clBlancheDalmond32', $FFFFEBCD ));
19850:     clBlueViolet32', $FF8A2BE2 ));
19851:     clBrown32', $FFA52A2A ));
19852:     clBurlyWood32', $FFDEB887 ));
19853:     clCadetblue32', $FF5F9EA0 ));
19854:     clChartReuse32', $FF7FFF00 ));
19855:     clChocolate32', $FFD2691E ));
19856:     clCoral32', $FFFF7F50 ));
19857:     clCornFlowerBlue32', $FF6495ED ));
19858:     clCornSilk32', $FFFFF8DC ));
19859:     clCrimson32', $FFDC143C ));
19860:     clDarkBlue32', $FF00008B ));
19861:     clDarkCyan32', $FF008B8B ));
19862:     clDarkGoldenRod32', $FFB8860B ));
19863:     clDarkGray32', $FFA9A9A9 ));
19864:     clDarkGreen32', $FF006400 ));
19865:     clDarkGrey32', $FFA9A9A9 ));
19866:     clDarkKhaki32', $FFBDB76B ));
19867:     clDarkMagenta32', $FF8B008B ));
19868:     clDarkOliveGreen32', $FF556B2F ));
19869:     clDarkOrange32', $FFFF8C00 ));
19870:     clDarkOrchid32', $FF9932CC ));
19871:     clDarkRed32', $FF8B0000 ));
19872:     clDarkSalmon32', $FFE9967A ));
19873:     clDarkSeaGreen32', $FF8FBC8F ));
19874:     clDarkSlateBlue32', $FF483D8B ));
19875:     clDarkSlateGray32', $FF2F4F4F ));
19876:     clDarkSlateGrey32', $FF2F4F4F ));
19877:     clDarkTurquoise32', $FF00CED1 ));
19878:     clDarkViolet32', $FF9400D3 ));
19879:     clDeepPink32', $FFFF1493 ));
19880:     clDeepSkyBlue32', $FF00BFFF ));
19881:     clDodgerBlue32', $FF1E90FF ));
19882:     clFireBrick32', $FFB22222 ));
19883:     clFloralWhite32', $FFFFFAF0 ));
19884:     clGainsBoro32', $FFDCDCDC ));
19885:     clGhostWhite32', $FFF8F8FF ));
19886:     clGold32', $FFFFD700 ));
19887:     clGoldenRod32', $FFDAA520 ));
19888:     clGreenYellow32', $FFADFF2F ));
19889:     clGrey32', $FF808080 ));
19890:     clHoneyDew32', $FFF0FFF0 ));
19891:     clHotPink32', $FFFF69B4 ));
19892:     clIndianRed32', $FFCD5C5C ));
19893:     clIndigo32', $FF4B0082 ));
19894:     clIvory32', $FFFFFFF0 ));
19895:     clKhaki32', $FFF0E68C ));
19896:     clLavender32', $FFE6E6FA ));
19897:     clLavenderBlush32', $FFFFF0F5 ));
19898:     clLawnGreen32', $FF7CFC00 ));
```

```
19899:     clLemonChiffon32', $FFFFFACD ));
19900:     clLightBlue32', $FFADD8E6 ));
19901:     clLightCoral32', $FFF08080 ));
19902:     clLightCyan32', $FFE0FFFF ));
19903:     clLightGoldenRodYellow32', $FFFAFAD2 ));
19904:     clLightGreen32', $FF90EE90 ));
19905:     clLightGrey32', $FFD3D3D3 ));
19906:     clLightPink32', $FFFFB6C1 ));
19907:     clLightSalmon32', $FFFFA07A ));
19908:     clLightSeagreen32', $FF20B2AA ));
19909:     clLightSkyblue32', $FF87CEFA ));
19910:     clLightSlategray32', $FF778899 ));
19911:     clLightSlategrey32', $FF778899 ));
19912:     clLightSteelblue32', $FFB0C4DE ));
19913:     clLightYellow32', $FFFFFFE0 ));
19914:     clLtGray32', $FFC0C0C0 ));
19915:     clMedGray32', $FFA0A0A4 ));
19916:     clDkGray32', $FF808080 ));
19917:     clMoneyGreen32', $FFC0DCC0 ));
19918:     clLegacySkyBlue32', $FFA6CAF0 ));
19919:     clCream32', $FFFFFBF0 ));
19920:     clLimeGreen32', $FF32CD32 ));
19921:     clLinen32', $FFFAF0E6 ));
19922:     clMediumAquamarine32', $FF66CDAA ));
19923:     clMediumBlue32', $FF0000CD ));
19924:     clMediumOrchid32', $FFBA55D3 ));
19925:     clMediumPurple32', $FF9370DB ));
19926:     clMediumSeaGreen32', $FF3CB371 ));
19927:     clMediumSlateBlue32', $FF7B68EE ));
19928:     clMediumSpringGreen32', $FF00FA9A ));
19929:     clMediumTurquoise32', $FF48D1CC ));
19930:     clMediumVioletRed32', $FFC71585 ));
19931:     clMidnightBlue32', $FF191970 ));
19932:     clMintCream32', $FFF5FFFA ));
19933:     clMistyRose32', $FFFFE4E1 ));
19934:     clMoccasin32', $FFFFE4B5 ));
19935:     clNavajoWhite32', $FFFFDEAD ));
19936:     clOldLace32', $FFFDF5E6 ));
19937:     clOliveDrab32', $FF6B8E23 ));
19938:     clOrange32', $FFFFA500 ));
19939:     clOrangeRed32', $FFFF4500 ));
19940:     clOrchid32', $FFDA70D6 ));
19941:     clPaleGoldenRod32', $FFEEE8AA ));
19942:     clPaleGreen32', $FF98FB98 ));
19943:     clPaleTurquoise32', $FFAFEEEE ));
19944:     clPaleVioletred32', $FFDB7093 ));
19945:     clPapayaWhip32', $FFFFEFD5 ));
19946:     clPeachPuff32', $FFFFDAB9 ));
19947:     clPeru32', $FFCD853F ));
19948:     clPlum32', $FFDDA0DD ));
19949:     clPowderBlue32', $FFB0E0E6 ));
19950:     clRosyBrown32', $FFBC8F8F ));
19951:     clRoyalBlue32', $FF4169E1 ));
19952:     clSaddleBrown32', $FF8B4513 ));
19953:     clSalmon32', $FFFA8072 ));
19954:     clSandyBrown32', $FFF4A460 ));
19955:     clSeaGreen32', $FF2E8B57 ));
19956:     clSeaShell32', $FFFFF5EE ));
19957:     clSienna32', $FFA0522D ));
19958:     clSilver32', $FFC0C0C0 ));
19959:     clSkyblue32', $FF87CEEB ));
19960:     clSlateBlue32', $FF6A5ACD ));
19961:     clSlateGray32', $FF708090 ));
19962:     clSlateGrey32', $FF708090 ));
19963:     clSnow32', $FFFFFAFA ));
19964:     clSpringgreen32', $FF00FF7F ));
19965:     clSteelblue32', $FF4682B4 ));
19966:     clTan32', $FFD2B48C ));
19967:     clThistle32', $FFD8BFD8 ));
19968:     clTomato32', $FFFF6347 ));
19969:     clTurquoise32', $FF40E0D0 ));
19970:     clViolet32', $FFEE82EE ));
19971:     clWheat32', $FFF5DEB3 ));
19972:     clWhitesmoke32', $FFF5F5F5 ));
19973:     clYellowgreen32', $FF9ACD32 ));
19974:     clTrWhite32', $7FFFFFFF ));
19975:     clTrBlack32', $7F000000 ));
19976:     clTrRed32', $7FFF0000 ));
19977:     clTrGreen32', $7F00FF00 ));
19978:     clTrBlue32', $7F0000FF ));
19979:   // Fixed point math constants
19980:   FixedOne = $10000;  FixedHalf = $7FFF;
19981:   FixedPI  = Round(PI * FixedOne);
19982:   FixedToFloat = 1/FixedOne;
19983:
19984:  Special Types
19985: ************************************************
19986:   type Complex = record
19987:     X, Y : Float;
```

```
19988:   end;
19989:   type TVector     = array of Float;
19990:   TIntVector  = array of Integer;
19991:   TCompVector = array of Complex;
19992:   TBoolVector = array of Boolean;
19993:   TStrVector  = array of String;
19994:   TMatrix     = array of TVector;
19995:   TIntMatrix  = array of TIntVector;
19996:   TCompMatrix = array of TCompVector;
19997:   TBoolMatrix = array of TBoolVector;
19998:   TStrMatrix  = array of TStrVector;
19999:   TByteArray = array[0..32767] of byte; !
20000:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
20001:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
20002:   T2StringArray = array of array of string;
20003:   T2IntegerArray = array of array of integer;
20004:   AddTypeS('INT_PTR', 'Integer
20005:   AddTypeS('LONG_PTR', 'Integer
20006:   AddTypeS('UINT_PTR', 'Cardinal
20007:   AddTypeS('ULONG_PTR', 'Cardinal
20008:   AddTypeS('DWORD_PTR', 'ULONG_PTR
20009:   TIntegerDynArray', 'array of Integer
20010:   TCardinalDynArray', 'array of Cardinal
20011:   TWordDynArray', 'array of Word
20012:   TSmallIntDynArray', 'array of SmallInt
20013:   TByteDynArray', 'array of Byte
20014:   TShortIntDynArray', 'array of ShortInt
20015:   TInt64DynArray', 'array of Int64
20016:   TLongWordDynArray', 'array of LongWord
20017:   TSingleDynArray', 'array of Single
20018:   TDoubleDynArray', 'array of Double
20019:   TBooleanDynArray', 'array of Boolean
20020:   TStringDynArray', 'array of string
20021:   TWideStringDynArray', 'array of WideString
20022:   TDynByteArray     = array of Byte;
20023:   TDynShortintArray = array of Shortint;
20024:   TDynSmallintArray = array of Smallint;
20025:   TDynWordArray     = array of Word;
20026:   TDynIntegerArray  = array of Integer;
20027:   TDynLongintArray  = array of Longint;
20028:   TDynCardinalArray = array of Cardinal;
20029:   TDynInt64Array    = array of Int64;
20030:   TDynExtendedArray = array of Extended;
20031:   TDynDoubleArray   = array of Double;
20032:   TDynSingleArray   = array of Single;
20033:   TDynFloatArray    = array of Float;
20034:   TDynPointerArray  = array of Pointer;
20035:   TDynStringArray   = array of string;
20036:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
20037:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
20038:   TSynSearchOptions = set of TSynSearchOption;
20039:
20040:
20041: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
20042: ----------------------------------------------------------------------------
20043: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
20044: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
20045: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
20046: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
20047: function CheckStringSum(vstring: string): integer;
20048: function HexToInt(HexNum: string): LongInt;
20049: function IntToBin(Int: Integer): String;
20050: function BinToInt(Binary: String): Integer;
20051: function HexToBin(HexNum: string): string; external2
20052: function BinToHex(Binary: String): string;
20053: function IntToFloat(i: Integer): double;
20054: function AddThousandSeparator(S: string; myChr: Char): string;
20055: function Max3(const X,Y,Z: Integer): Integer;
20056: procedure Swap(var X,Y: char); // faster without inline
20057: procedure ReverseString(var S: String);
20058: function CharToHexStr(Value: Char): string;
20059: function CharToUniCode(Value: Char): string;
20060: function Hex2Dec(Value: Str002): Byte;
20061: function HexStrCodeToStr(Value: string): string;
20062: function HexToStr(i: integer; value: string): string;
20063: function UniCodeToStr(Value: string): string;
20064: function CRC16(statement: string): string;
20065: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
20066: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
20067: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
20068: Procedure ExecuteCommand(executeFile, paramstring: string);
20069: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
20070: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
20071: procedure SearchAndOpenDoc(vfilenamepath: string);
20072: procedure ShowInterfaces(myFile: string);
20073: function Fact2(av: integer): extended;
20074: Function BoolToStr(B: Boolean): string;
20075: Function GCD(x, y : LongInt) : LongInt;
20076: function LCM(m,n: longint): longint;
```

```
20077: function GetASCII: string;
20078: function GetItemHeight(Font: TFont): Integer;
20079: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
20080: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
20081: function getHINSTANCE: longword;
20082: function getHMODULE: longword;
20083: function GetASCII: string;
20084: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
20085: function WordIsOk(const AWord: string; var VW: Word): boolean;
20086: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
20087: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
20088: function SafeStr(const s: string): string;
20089: function ExtractUrlPath(const FileName: string): string;
20090: function ExtractUrlName(const FileName: string): string;
20091: function IsInternet: boolean;
20092: function RotateLeft1Bit_u32( Value: uint32): uint32;
20093: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats: Bool);
20094: procedure getEnvironmentInfo;
20095: procedure AntiFreeze;
20096: function GetCPUSpeed: Double;
20097: function IsVirtualPcGuest : Boolean;
20098: function IsVmWareGuest : Boolean;
20099: procedure StartSerialDialog;
20100: function IsWoW64: boolean;
20101: function IsWow64String(var s: string): Boolean;
20102: procedure StartThreadDemo;
20103: Function RGB(R,G,B: Byte): TColor;
20104: Function Sendln(amess: string): boolean;
20105: Procedure maXbox;
20106: Function AspectRatio(aWidth, aHeight: Integer): String;
20107: function wget(aURL, afile: string): boolean;
20108: procedure PrintList(Value: TStringList);
20109: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
20110: procedure getEnvironmentInfo;
20111: procedure AntiFreeze;
20112: function getBitmap(apath: string): TBitmap;
20113: procedure ShowMessageBig(const aText : string);
20114: function YesNoDialog(const ACaption, AMsg: string): boolean;
20115: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
20116: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
20117: //function myStrToBytes(const Value: String): TBytes;
20118: //function myBytesToStr(const Value: TBytes): String;
20119: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20120: function getBitmap(apath: string): TBitmap;
20121: procedure ShowMessageBig(const aText : string);
20122: Function StrToBytes(const Value: String): TBytes;
20123: Function BytesToStr(const Value: TBytes): String;
20124: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
20125: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
20126: function FindInPaths(const fileName, paths : String) : String;
20127: procedure initHexArray(var hexn: THexArray);
20128: function josephusG(n,k: integer; var graphout: string): integer;
20129: function isPowerof2(num: int64): boolean;
20130: function powerOf2(exponent: integer): int64;
20131: function getBigPI: string;
20132: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
20133:  function GetASCIILine: string;
20134: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
20135:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
20136: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
20137: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
20138: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
20139: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
20140: function isKeypressed: boolean;
20141: function Keypress: boolean;
20142: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
20143: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
20144: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
20145: function GetOSName: string;
20146: function GetOSVersion: string;
20147: function GetOSNumber: string;
20148: function getEnvironmentString: string;
20149: procedure StrReplace(var Str: String; Old, New: String);
20150: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
20151: function getTeamViewerID: string;
20152: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
20153: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
20154: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
20155: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
20156: function StartSocketService: Boolean;
20157: procedure StartSocketServiceForm;
20158: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
20159: function GetFileList1(apath: string): TStringlist;
20160: procedure LetFileList(FileList: TStringlist; apath: string);
20161: procedure StartWeb(aurl: string);
20162: function GetTodayFiles(startdir, amask: string): TStringlist;
20163: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
```

```
20164: function JavahashCode(val: string): Integer;
20165: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
20166: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
20167: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
20168: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
20169: Procedure ConvertToGray(Cnv: TCanvas);
20170: function GetFileDate(aFile:string; aWithTime:Boolean):string;
20171: procedure ShowMemory;
20172: function ShowMemory2: string;
20173: function getHostIP: string;
20174: procedure ShowBitmap(bmap: TBitmap);
20175: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
20176: function CreateDBGridForm(dblist: TStringList): TListbox;
20177: function isService: boolean;
20178: function isApplication: boolean;
20179: function isTerminalSession: boolean;
20180:
20181:
20182: // News of 3.9.8 up
20183: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20184: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20185: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20186: JvChart - TJvChart Component - 2009 Public
20187: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
20188: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
20189: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
20190: DMath DLL included incl. Demos
20191: Interface Navigator menu/View/Intf Navigator
20192: Unit Explorer menu/Debug/Units Explorer
20193: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
20194: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
20195: Script History to 9 Files WebServer light /Options/Addons/WebServer
20196: Full Text Finder, JVSimLogic Simulator Package
20197: Halt-Stop Program in Menu, WebServer2, Stop Event ,
20198: Conversion Routines, Prebuild Forms, CodeSearch
20199: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
20200: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
20201: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
20202: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
20203: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
20204: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
20205: IDE Reflection API, Session Service Shell S3
20206: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
20207: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
20208: arduino map() function, PMRandom Generator
20209: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
20210: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
20211: REST Test Lib, Multilang Component, Forth Interpreter
20212: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
20213: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
20214: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20215: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
20216: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
20217: QRCode Service, add more CFunctions like CDateTime of Synapse
20218: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
20219: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
20220: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
20221: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
20222: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
20223: BOLD Package, Indy Package5, maTRIx. MATHEMAX
20224: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
20225: emax layers: system-package-component-unit-class-function-block
20226: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
20227: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
20228: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
20229: OpenGL Game Demo: ..Options/Add Ons/Reversi
20230: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
20231: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
20232: 7% performance gain (hot spot profiling)
20233: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
20234: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
20235: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
20236: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
20237:
20238: add routines in 3.9.7.5
20239: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
20240: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
20241: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
20242: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
20243: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
20244: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
20245: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
20246:
20247: ////////////////////// TestUnits ////////////////////////
20248:   SelftestPEM;
20249:   SelfTestCFundamentUtils;
20250:   SelfTestCFileUtils;
20251:   SelfTestCDateTime;
20252:   SelfTestCTimer;
```

```
20253:   SelfTestCRandom;
20254:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
20255:            Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
20256:
20257:   // Note: There's no need for installing a client certificate in the
20258:   //       webbrowser. The server asks the webbrowser to send a certificate but
20259:   //       if nothing is installed the software will work because the server
20260:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
20261:   //       file: c_cacert.p12  password: c_cakey
20262:
20263:   TGraphicControl = class(TControl)
20264:   private
20265:     FCanvas: TCanvas;
20266:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20267:   protected
20268:     procedure Paint; virtual;
20269:     property Canvas: TCanvas read FCanvas;
20270:   public
20271:     constructor Create(AOwner: TComponent); override;
20272:     destructor Destroy; override;
20273:   end;
20274:
20275:   TCustomControl = class(TWinControl)
20276:   private
20277:     FCanvas: TCanvas;
20278:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
20279:   protected
20280:     procedure Paint; virtual;
20281:     procedure PaintWindow(DC: HDC); override;
20282:     property Canvas: TCanvas read FCanvas;
20283:   public
20284:     constructor Create(AOwner: TComponent); override;
20285:     destructor Destroy; override;
20286:   end;
20287:     RegisterPublishedProperties;
20288:     ('ONCHANGE', 'TNotifyEvent', iptrw);
20289:     ('ONCLICK', 'TNotifyEvent', iptrw);
20290:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
20291:     ('ONENTER', 'TNotifyEvent', iptrw);
20292:     ('ONEXIT', 'TNotifyEvent', iptrw);
20293:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
20294:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
20295:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
20296:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
20297:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
20298: //*************************************************************************
20299:   // To stop the while loop, click on Options/Show Include (boolean switch)!
20300:   Control a loop in a script with a form event:
20301:   IncludeON;   //control the while loop
20302:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
20303:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
20304:
20305: //-------------------------------------------------------------------------
20306: //**************mX4 ini-file Configuration***********************************
20307: //-------------------------------------------------------------------------
20308:   using config file maxboxdef.ini       menu/Help/Config File
20309:
20310: //*** Definitions for maXbox mX3 ***
20311: [FORM]
20312: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
20313: FONTSIZE=14
20314: EXTENSION=txt
20315: SCREENX=1386
20316: SCREENY=1077
20317: MEMHEIGHT=350
20318: PRINTFONT=Courier New //GUI Settings
20319: LINENUMBERS=Y   //line numbers at gutter in editor at left side
20320: EXCEPTIONLOG=Y  //store excepts + success in 2 log files see below! -menu Debug/Show Last Exceptions
20321: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
20322: BOOTSCRIPT=Y    //enabling load a boot script
20323: MEMORYREPORT=Y  //shows memory report on closing maXbox
20324: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
20325: NAVIGATOR=N     //shows function list at the right side of editor
20326: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
20327: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
20328: [WEB]
20329: IPPORT=8080     //for internal webserver - menu /Options/Add Ons/WebServer
20330: IPHOST=192.168.1.53
20331: ROOTCERT=filepathY
20332: SCERT=filepathY
20333: RSAKEY=filepathY
20334: VERSIONCHECK=Y
20335:
20336: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
20337: Also possible to set report memory in script to override ini setting
20338: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
20339:
20340: After Change the ini file you can reload the file with ../Help/Config Update
20341:
```

```
20342: //-------------------------------------------------------------------------------
20343: //**************mX4 maildef.ini ini-file Configuration*********************
20344: //-------------------------------------------------------------------------------
20345: //*** Definitions for maXMail ***
20346: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
20347: [MAXMAIL]
20348: HOST=getmail.softwareschule.ch
20349: USER=mailusername
20350: PASS=password
20351: PORT=110
20352: SSL=Y
20353: BODY=Y
20354: LAST=5
20355:
20356: ADO Connection String:
20357: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
20358: \452_dbtreeview2access.txt
20359: program TestDbTreeViewMainForm2_ACCESS;
20360:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
20361:                    +Exepath+'\examples\detail.mdb;Persist Security Info=False';
20362:
20363: OpenSSL Lib:  unit ssl_openssl_lib;
20364: {$IFDEF CIL}
20365: const
20366:   {$IFDEF LINUX}
20367:   DLLSSLName = 'libssl.so';
20368:   DLLUtilName = 'libcrypto.so';
20369:   {$ELSE}
20370:   DLLSSLName = 'ssleay32.dll';
20371:   DLLUtilName = 'libeay32.dll';
20372:   {$ENDIF}
20373: {$ELSE}
20374: var
20375:   {$IFNDEF MSWINDOWS}
20376:     {$IFDEF DARWIN}
20377:     DLLSSLName: string = 'libssl.dylib';
20378:     DLLUtilName: string = 'libcrypto.dylib';
20379:     {$ELSE}
20380:     DLLSSLName: string = 'libssl.so';
20381:     DLLUtilName: string = 'libcrypto.so';
20382:     {$ENDIF}
20383:   {$ELSE}
20384:   DLLSSLName: string = 'ssleay32.dll';
20385:   DLLSSLName2: string = 'libssl32.dll';
20386:   DLLUtilName: string = 'libeay32.dll';
20387:   {$ENDIF}
20388: {$ENDIF}
20389:
20390:
20391: //-------------------------------------------------------------------------------
20392: //**************mX4 Macro Tags  ***********************************************
20393: //-------------------------------------------------- ----------------------
20394:
20395:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
       E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
20396:
20397: //Tag Macros
20398:
20399:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
20400:
20401: //Tag Macros
20402: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
20403: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
20404: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
20405: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
20406: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
20407: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
20408: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
20409: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
20410: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
20411:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
20412: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
20413: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20414:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20415: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
20416:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
20417: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
20418:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
20419:
20420: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
20421:
20422: //Replace Macros
20423:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
20424:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
20425:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
20426:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
20427:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
20428:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
20429:
```

```
20430:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20431:       [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
20432: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20433:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
20434:
20435: //-------------------------------------------------------------------------
20436: //**************mX4 ToDo List Tags  ../Help/ToDo List************************
20437: //-------------------------------------------------------- ---------------------
20438:
20439:     while I < sl.Count do begin
20440: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
20441:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
20442:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
20443:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
20444:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
20445:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
20446:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
20447:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
20448:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
20449:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
20450:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
20451:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
20452:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
20453:       Inc(I);
20454:     end;
20455:
20456: //-------------------------------------------------------------------------
20457: //**************mX4 Public  Tools API ****************************************
20458: //-------------------------------------------------------------------------
20459:   file : unit uPSI_fMain.pas;              {$OTAP} Open Tools API Catalog
20460:  // Those functions concern the editor and preprocessor, all of the IDE
20461:  Example: Call it with maxform1.Info1Click(self)
20462:  Note: Call all Methods with maxForm1., e.g.:
20463:                       maxForm1.ShellStyle1Click(self);
20464:
20465: procedure SIRegister_fMain(CL: TPSPascalCompiler);
20466: begin
20467:  Const('BYTECODE','String 'bytecode.txt'
20468:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
20469:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
20470:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
20471:  Const('PSINC','String PS Includes (*.inc)|*.INC
20472:  Const('DEFFILENAME','String 'firstdemo.txt
20473:  Const('DEFINIFILE','String 'maxboxdef.ini
20474:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
20475:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
20476:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
20477:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
20478:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
20479:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
20480:  Const('INCLUDEBOX','String 'pas_includebox.inc
20481:  Const('BOOTSCRIPT','String 'maxbootscript.txt
20482:  Const('MBVERSION','String '3.9.9.101
20483:  Const('VERSION','String'3.9.9.101
20484:  Const('MBVER','String '399
20485:  Const('MBVERI','Integer'(399);
20486:  Const('MBVERIALL','Integer'(399101);
20487:  Const('EXENAME','String 'maXbox3.exe
20488:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
20489:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
20490:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
20491:  Const('MXINTERNETCHECK','String 'www.ask.com
20492:  Const('MXMAIL','String 'max@kleiner.com
20493:  Const('TAB','Char #$09);
20494:  Const('CODECOMPLETION','String 'bds_delphi.dci
20495:  SIRegister_TMaxForm1(CL);
20496: end;
20497:
20498:   with FindClass('TForm'),'TMaxForm1') do begin
20499:     memo2', 'TMemo', iptrw);
20500:     memo1', 'TSynMemo', iptrw);
20501:     CB1SCList', 'TComboBox', iptrw);
20502:     mxNavigator', 'TComboBox', iptrw);
20503:     IPHost', 'string', iptrw);
20504:     IPPort', 'integer', iptrw);
20505:     COMPort', 'integer', iptrw);      //3.9.6.4
20506:     Splitter1', 'TSplitter', iptrw);
20507:     PSScript', 'TPSScript', iptrw);
20508:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
20509:     MainMenu1', 'TMainMenu', iptrw);
20510:     Program1', 'TMenuItem', iptrw);
20511:     Compile1', 'TMenuItem', iptrw);
20512:     Files1', 'TMenuItem', iptrw);
20513:     open1', 'TMenuItem', iptrw);
20514:     Save2', 'TMenuItem', iptrw);
20515:     Options1', 'TMenuItem', iptrw);
20516:     Savebefore1', 'TMenuItem', iptrw);
20517:     Largefont1', 'TMenuItem', iptrw);
20518:     sBytecode1', 'TMenuItem', iptrw);
```

```
20519:     Saveas3', 'TMenuItem', iptrw);
20520:     Clear1', 'TMenuItem', iptrw);
20521:     Slinenumbers1', 'TMenuItem', iptrw);
20522:     About1', 'TMenuItem', iptrw);
20523:     Search1', 'TMenuItem', iptrw);
20524:     SynPasSyn1', 'TSynPasSyn', iptrw);
20525:     memo1', 'TSynMemo', iptrw);
20526:     SynEditSearch1', 'TSynEditSearch', iptrw);
20527:     WordWrap1', 'TMenuItem', iptrw);
20528:     XPManifest1', 'TXPManifest', iptrw);
20529:     SearchNext1', 'TMenuItem', iptrw);
20530:     Replace1', 'TMenuItem', iptrw);
20531:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
20532:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
20533:     ShowInclude1', 'TMenuItem', iptrw);
20534:     SynEditPrint1', 'TSynEditPrint', iptrw);
20535:     Printout1', 'TMenuItem', iptrw);
20536:     mnPrintColors1', 'TMenuItem', iptrw);
20537:     dlgFilePrint', 'TPrintDialog', iptrw);
20538:     dlgPrintFont1', 'TFontDialog', iptrw);
20539:     mnuPrintFont1', 'TMenuItem', iptrw);
20540:     Include1', 'TMenuItem', iptrw);
20541:     CodeCompletionList1', 'TMenuItem', iptrw);
20542:     IncludeList1', 'TMenuItem', iptrw);
20543:     ImageList1', 'TImageList', iptrw);
20544:     ImageList2', 'TImageList', iptrw);
20545:     CoolBar1', 'TCoolBar', iptrw);
20546:     ToolBar1', 'TToolBar', iptrw);
20547:     tbtnLoad', 'TToolButton', iptrw);
20548:     ToolButton2', 'TToolButton', iptrw);
20549:     tbtnFind', 'TToolButton', iptrw);
20550:     tbtnCompile', 'TToolButton', iptrw);
20551:     tbtnTrans', 'TToolButton', iptrw);
20552:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
20553:     toolbtnTutorial', 'TToolButton', iptrw);
20554:     tbtn6res', 'TToolButton', iptrw);
20555:     ToolButton5', 'TToolButton', iptrw);
20556:     ToolButton1', 'TToolButton', iptrw);
20557:     ToolButton3', 'TToolButton', iptrw);
20558:     statusBar1', 'TStatusBar', iptrw);
20559:     SaveOutput1', 'TMenuItem', iptrw);
20560:     ExportClipboard1', 'TMenuItem', iptrw);
20561:     Close1', 'TMenuItem', iptrw);
20562:     Manual1', 'TMenuItem', iptrw);
20563:     About2', 'TMenuItem', iptrw);
20564:     loadLastfile1', 'TMenuItem', iptrw);
20565:     imglogo', 'TImage', iptrw);
20566:     cedebug', 'TPSScriptDebugger', iptrw);
20567:     debugPopupMenu1', 'TPopupMenu', iptrw);
20568:     BreakPointMenu', 'TMenuItem', iptrw);
20569:     Decompile1', 'TMenuItem', iptrw);
20570:     StepInto1', 'TMenuItem', iptrw);
20571:     StepOut1', 'TMenuItem', iptrw);
20572:     Reset1', 'TMenuItem', iptrw);
20573:     DebugRun1', 'TMenuItem', iptrw);
20574:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
20575:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20576:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
20577:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20578:     tutorial4', 'TMenuItem', iptrw);
20579:     ExporttoClipboard1', 'TMenuItem', iptrw);
20580:     ImportfromClipboard1', 'TMenuItem', iptrw);
20581:     N4', 'TMenuItem', iptrw);
20582:     N5', 'TMenuItem', iptrw);
20583:     N6', 'TMenuItem', iptrw);
20584:     ImportfromClipboard2', 'TMenuItem', iptrw);
20585:     tutorial1', 'TMenuItem', iptrw);
20586:     N7', 'TMenuItem', iptrw);
20587:     ShowSpecChars1', 'TMenuItem', iptrw);
20588:     OpenDirectory1', 'TMenuItem', iptrw);
20589:     procMess', 'TMenuItem', iptrw);
20590:     tbtnUseCase', 'TToolButton', iptrw);
20591:     ToolButton7', 'TToolButton', iptrw);
20592:     EditFont1', 'TMenuItem', iptrw);
20593:     UseCase1', 'TMenuItem', iptrw);
20594:     tutorial21', 'TMenuItem', iptrw);
20595:     OpenUseCase1', 'TMenuItem', iptrw);
20596:     PSImport_DB1', 'TPSImport_DB', iptrw);
20597:     tutorial31', 'TMenuItem', iptrw);
20598:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
20599:     HTMLSyntax1', 'TMenuItem', iptrw);
20600:     ShowInterfaces1', 'TMenuItem', iptrw);
20601:     Tutorial5', 'TMenuItem', iptrw);
20602:     AllFunctionsList1', 'TMenuItem', iptrw);
20603:     ShowLastException1', 'TMenuItem', iptrw);
20604:     PlayMP31', 'TMenuItem', iptrw);
20605:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
20606:     texSyntax1', 'TMenuItem', iptrw);
20607:     N8', 'TMenuItem', iptrw);
```

```
20608:     GetEMails1', 'TMenuItem', iptrw);
20609:     SynCppSyn1', 'TSynCppSyn', iptrw);
20610:     CSyntax1', 'TMenuItem', iptrw);
20611:     Tutorial6', 'TMenuItem', iptrw);
20612:     New1', 'TMenuItem', iptrw);
20613:     AllObjectsList1', 'TMenuItem', iptrw);
20614:     LoadBytecode1', 'TMenuItem', iptrw);
20615:     CipherFile1', 'TMenuItem', iptrw);
20616:     N9', 'TMenuItem', iptrw);
20617:     N10', 'TMenuItem', iptrw);
20618:     Tutorial11', 'TMenuItem', iptrw);
20619:     Tutorial71', 'TMenuItem', iptrw);
20620:     UpdateService1', 'TMenuItem', iptrw);
20621:     PascalSchool1', 'TMenuItem', iptrw);
20622:     Tutorial81', 'TMenuItem', iptrw);
20623:     DelphiSite1', 'TMenuItem', iptrw);
20624:     Output1', 'TMenuItem', iptrw);
20625:     TerminalStyle1', 'TMenuItem', iptrw);
20626:     ReadOnly1', 'TMenuItem', iptrw);
20627:     ShellStyle1', 'TMenuItem', iptrw);
20628:     BigScreen1', 'TMenuItem', iptrw);
20629:     Tutorial91', 'TMenuItem', iptrw);
20630:     SaveOutput2', 'TMenuItem', iptrw);
20631:     N11', 'TMenuItem', iptrw);
20632:     SaveScreenshot', 'TMenuItem', iptrw);
20633:     Tutorial101', 'TMenuItem', iptrw);
20634:     SQLSyntax1', 'TMenuItem', iptrw);
20635:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
20636:     Console1', 'TMenuItem', iptrw);
20637:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
20638:     XMLSyntax1', 'TMenuItem', iptrw);
20639:     ComponentCount1', 'TMenuItem', iptrw);
20640:     NewInstance1', 'TMenuItem', iptrw);
20641:     toolbtnTutorial', 'TToolButton', iptrw);
20642:     Memory1', 'TMenuItem', iptrw);
20643:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
20644:     JavaSyntax1', 'TMenuItem', iptrw);
20645:     SyntaxCheck1', 'TMenuItem', iptrw);
20646:     Tutorial10Statistics1', 'TMenuItem', iptrw);
20647:     ScriptExplorer1', 'TMenuItem', iptrw);
20648:     FormOutput1', 'TMenuItem', iptrw);
20649:     ArduinoDump1', 'TMenuItem', iptrw);
20650:     AndroidDump1', 'TMenuItem', iptrw);
20651:     GotoEnd1', 'TMenuItem', iptrw);
20652:     AllResourceList1', 'TMenuItem', iptrw);
20653:     ToolButton4', 'TToolButton', iptrw);
20654:     tbtn6res', 'TToolButton', iptrw);
20655:     Tutorial11Forms1', 'TMenuItem', iptrw);
20656:     Tutorial12SQL1', 'TMenuItem', iptrw);
20657:     ResourceExplore1', 'TMenuItem', iptrw);
20658:     Info1', 'TMenuItem', iptrw);
20659:     N12', 'TMenuItem', iptrw);
20660:     CryptoBox1', 'TMenuItem', iptrw);
20661:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
20662:     CipherFile2', 'TMenuItem', iptrw);
20663:     N13', 'TMenuItem', iptrw);
20664:     ModulesCount1', 'TMenuItem', iptrw);
20665:     AddOns2', 'TMenuItem', iptrw);
20666:     N4GewinntGame1', 'TMenuItem', iptrw);
20667:     DocuforAddOns1', 'TMenuItem', iptrw);
20668:     Tutorial14Async1', 'TMenuItem', iptrw);
20669:     Lessons15Review1', 'TMenuItem', iptrw);
20670:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
20671:     PHPSyntax1', 'TMenuItem', iptrw);
20672:     Breakpoint1', 'TMenuItem', iptrw);
20673:     SerialRS2321', 'TMenuItem', iptrw);
20674:     N14', 'TMenuItem', iptrw);
20675:     SynCSSyn1', 'TSynCSSyn', iptrw);
20676:     CSyntax2', 'TMenuItem', iptrw);
20677:     Calculator1', 'TMenuItem', iptrw);
20678:     tbtnSerial', 'TToolButton', iptrw);
20679:     ToolButton8', 'TToolButton', iptrw);
20680:     Tutorial151', 'TMenuItem', iptrw);
20681:     N15', 'TMenuItem', iptrw);
20682:     N16', 'TMenuItem', iptrw);
20683:     ControlBar1', 'TControlBar', iptrw);
20684:     ToolBar2', 'TToolBar', iptrw);
20685:     BtnOpen', 'TToolButton', iptrw);
20686:     BtnSave', 'TToolButton', iptrw);
20687:     BtnPrint', 'TToolButton', iptrw);
20688:     BtnColors', 'TToolButton', iptrw);
20689:     btnClassReport', 'TToolButton', iptrw);
20690:     BtnRotateRight', 'TToolButton', iptrw);
20691:     BtnFullSize', 'TToolButton', iptrw);
20692:     BtnFitToWindowSize', 'TToolButton', iptrw);
20693:     BtnZoomMinus', 'TToolButton', iptrw);
20694:     BtnZoomPlus', 'TToolButton', iptrw);
20695:     Panel1', 'TPanel', iptrw);
20696:     LabelBrettgroesse', 'TLabel', iptrw);
```

```
20697:     CB1SCList', 'TComboBox', iptrw);
20698:     ImageListNormal', 'TImageList', iptrw);
20699:     spbtnexplore', 'TSpeedButton', iptrw);
20700:     spbtnexample', 'TSpeedButton', iptrw);
20701:     spbsaveas', 'TSpeedButton', iptrw);
20702:     imglogobox', 'TImage', iptrw);
20703:     EnlargeFont1', 'TMenuItem', iptrw);
20704:     EnlargeFont2', 'TMenuItem', iptrw);
20705:     ShrinkFont1', 'TMenuItem', iptrw);
20706:     ThreadDemo1', 'TMenuItem', iptrw);
20707:     HEXEditor1', 'TMenuItem', iptrw);
20708:     HEXView1', 'TMenuItem', iptrw);
20709:     HEXInspect1', 'TMenuItem', iptrw);
20710:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
20711:     ExporttoHTML1', 'TMenuItem', iptrw);
20712:     ClassCount1', 'TMenuItem', iptrw);
20713:     HTMLOutput1', 'TMenuItem', iptrw);
20714:     HEXEditor2', 'TMenuItem', iptrw);
20715:     Minesweeper1', 'TMenuItem', iptrw);
20716:     N17', 'TMenuItem', iptrw);
20717:     PicturePuzzle1', 'TMenuItem', iptrw);
20718:     sbvclhelp', 'TSpeedButton', iptrw);
20719:     DependencyWalker1', 'TMenuItem', iptrw);
20720:     WebScanner1', 'TMenuItem', iptrw);
20721:     View1', 'TMenuItem', iptrw);
20722:     mnToolbar1', 'TMenuItem', iptrw);
20723:     mnStatusbar2', 'TMenuItem', iptrw);
20724:     mnConsole2', 'TMenuItem', iptrw);
20725:     mnCoolbar2', 'TMenuItem', iptrw);
20726:     mnSplitter2', 'TMenuItem', iptrw);
20727:     WebServer1', 'TMenuItem', iptrw);
20728:     Tutorial17Server1', 'TMenuItem', iptrw);
20729:     Tutorial18Arduino1', 'TMenuItem', iptrw);
20730:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
20731:     PerlSyntax1', 'TMenuItem', iptrw);
20732:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
20733:     PythonSyntax1', 'TMenuItem', iptrw);
20734:     DMathLibrary1', 'TMenuItem', iptrw);
20735:     IntfNavigator1', 'TMenuItem', iptrw);
20736:     EnlargeFontConsole1', 'TMenuItem', iptrw);
20737:     ShrinkFontConsole1', 'TMenuItem', iptrw);
20738:     SetInterfaceList1', 'TMenuItem', iptrw);
20739:     popintfList', 'TPopupMenu', iptrw);
20740:     intfAdd1', 'TMenuItem', iptrw);
20741:     intfDelete1', 'TMenuItem', iptrw);
20742:     intfRefactor1', 'TMenuItem', iptrw);
20743:     Defactor1', 'TMenuItem', iptrw);
20744:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
20745:     Tutorial20Regex', 'TMenuItem', iptrw);
20746:     N18', 'TMenuItem', iptrw);
20747:     ManualE1', 'TMenuItem', iptrw);
20748:     FullTextFinder1', 'TMenuItem', iptrw);
20749:     Move1', 'TMenuItem', iptrw);
20750:     FractalDemo1', 'TMenuItem', iptrw);
20751:     Tutorial21Android1', 'TMenuItem', iptrw);
20752:     Tutorial0Function1', 'TMenuItem', iptrw);
20753:     SimuLogBox1', 'TMenuItem', iptrw);
20754:     OpenExamples1', 'TMenuItem', iptrw);
20755:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
20756:     JavaScriptSyntax1', 'TMenuItem', iptrw);
20757:     Halt1', 'TMenuItem', iptrw);
20758:     CodeSearch1', 'TMenuItem', iptrw);
20759:     SynRubySyn1', 'TSynRubySyn', iptrw);
20760:     RubySyntax1', 'TMenuItem', iptrw);
20761:     Undo1', 'TMenuItem', iptrw);
20762:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
20763:     LinuxShellScript1', 'TMenuItem', iptrw);
20764:     Rename1', 'TMenuItem', iptrw);
20765:     spdcodesearch', 'TSpeedButton', iptrw);
20766:     Preview1', 'TMenuItem', iptrw);
20767:     Tutorial22Services1', 'TMenuItem', iptrw);
20768:     Tutorial23RealTime1', 'TMenuItem', iptrw);
20769:     Configuration1', 'TMenuItem', iptrw);
20770:     MP3Player1', 'TMenuItem', iptrw);
20771:     DLLSpy1', 'TMenuItem', iptrw);
20772:     SynURIOpener1', 'TSynURIOpener', iptrw);
20773:     SynURISyn1', 'TSynURISyn', iptrw);
20774:     URILinksClicks1', 'TMenuItem', iptrw);
20775:     EditReplace1', 'TMenuItem', iptrw);
20776:     GotoLine1', 'TMenuItem', iptrw);
20777:     ActiveLineColor1', 'TMenuItem', iptrw);
20778:     ConfigFile1', 'TMenuItem', iptrw);
20779:     Sort1Intflist', 'TMenuItem', iptrw);
20780:     Redo1', 'TMenuItem', iptrw);
20781:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
20782:     Tutorial25Configuration1', 'TMenuItem', iptrw);
20783:     IndentSelection1', 'TMenuItem', iptrw);
20784:     UnindentSection1', 'TMenuItem', iptrw);
20785:     SkyStyle1', 'TMenuItem', iptrw);
```

```
20786:     N19', 'TMenuItem', iptrw);
20787:     CountWords1', 'TMenuItem', iptrw);
20788:     imbookmarkimages', 'TImageList', iptrw);
20789:     Bookmark11', 'TMenuItem', iptrw);
20790:     N20', 'TMenuItem', iptrw);
20791:     Bookmark21', 'TMenuItem', iptrw);
20792:     Bookmark31', 'TMenuItem', iptrw);
20793:     Bookmark41', 'TMenuItem', iptrw);
20794:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
20795:
20796:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
20797:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
20798:     Procedure PSScriptCompile( Sender : TPSScript)
20799:     Procedure Compile1Click( Sender : TObject)
20800:     Procedure PSScriptExecute( Sender : TPSScript)
20801:     Procedure open1Click( Sender : TObject)
20802:     Procedure Save2Click( Sender : TObject)
20803:     Procedure Savebefore1Click( Sender : TObject)
20804:     Procedure Largefont1Click( Sender : TObject)
20805:     Procedure FormActivate( Sender : TObject)
20806:     Procedure SBytecode1Click( Sender : TObject)
20807:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
20808:     Procedure Saveas3Click( Sender : TObject)
20809:     Procedure Clear1Click( Sender : TObject)
20810:     Procedure Slinenumbers1Click( Sender : TObject)
20811:     Procedure About1Click( Sender : TObject)
20812:     Procedure Search1Click( Sender : TObject)
20813:     Procedure FormCreate( Sender : TObject)
20814:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
20815:                                                 var Action : TSynReplaceAction)
20816:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
20817:     Procedure WordWrap1Click( Sender : TObject)
20818:     Procedure SearchNext1Click( Sender : TObject)
20819:     Procedure Replace1Click( Sender : TObject)
20820:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
20821:     Procedure ShowInclude1Click( Sender : TObject)
20822:     Procedure Printout1Click( Sender : TObject)
20823:     Procedure mnuPrintFont1Click( Sender : TObject)
20824:     Procedure Include1Click( Sender : TObject)
20825:     Procedure FormDestroy( Sender : TObject)
20826:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
20827:     Procedure UpdateView1Click( Sender : TObject)
20828:     Procedure CodeCompletionList1Click( Sender : TObject)
20829:     Procedure SaveOutput1Click( Sender : TObject)
20830:     Procedure ExportClipboard1Click( Sender : TObject)
20831:     Procedure Close1Click( Sender : TObject)
20832:     Procedure Manual1Click( Sender : TObject)
20833:     Procedure LoadLastFile1Click( Sender : TObject)
20834:     Procedure Memo1Change( Sender : TObject)
20835:     Procedure Decompile1Click( Sender : TObject)
20836:     Procedure StepInto1Click( Sender : TObject)
20837:     Procedure StepOut1Click( Sender : TObject)
20838:     Procedure Reset1Click( Sender : TObject)
20839:     Procedure cedebugAfterExecute( Sender : TPSScript)
20840:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
20841:     Procedure cedebugCompile( Sender : TPSScript)
20842:     Procedure cedebugExecute( Sender : TPSScript)
20843:     Procedure cedebugIdle( Sender : TObject)
20844:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
20845:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
20846:     Procedure BreakPointMenuClick( Sender : TObject)
20847:     Procedure DebugRun1Click( Sender : TObject)
20848:     Procedure tutorial4Click( Sender : TObject)
20849:     Procedure ImportfromClipboard1Click( Sender : TObject)
20850:     Procedure ImportfromClipboard2Click( Sender : TObject)
20851:     Procedure tutorial1Click( Sender : TObject)
20852:     Procedure ShowSpecChars1Click( Sender : TObject)
20853:     Procedure StatusBar1DblClick( Sender : TObject)
20854:     Procedure PSScriptLine( Sender : TObject)
20855:     Procedure OpenDirectory1Click( Sender : TObject)
20856:     Procedure procMessClick( Sender : TObject)
20857:     Procedure tbtnUseCaseClick( Sender : TObject)
20858:     Procedure EditFont1Click( Sender : TObject)
20859:     Procedure tutorial21Click( Sender : TObject)
20860:     Procedure tutorial31Click( Sender : TObject)
20861:     Procedure HTMLSyntax1Click( Sender : TObject)
20862:     Procedure ShowInterfaces1Click( Sender : TObject)
20863:     Procedure Tutorial5Click( Sender : TObject)
20864:     Procedure ShowLastException1Click( Sender : TObject)
20865:     Procedure PlayMP31Click( Sender : TObject)
20866:     Procedure AllFunctionsList1Click( Sender : TObject)
20867:     Procedure texSyntax1Click( Sender : TObject)
20868:     Procedure GetEMails1Click( Sender : TObject)
20869:     procedure DelphiSite1Click(Sender: TObject);
20870:     procedure TerminalStyle1Click(Sender: TObject);
20871:     procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
20872:     procedure ShellStyle1Click(Sender: TObject);
20873:     procedure Console1Click(Sender: TObject);     //3.2
20874:     procedure BigScreen1Click(Sender: TObject);
```

```
20875:     procedure Tutorial91Click(Sender: TObject);
20876:     procedure SaveScreenshotClick(Sender: TObject);
20877:     procedure Tutorial101Click(Sender: TObject);
20878:     procedure SQLSyntax1Click(Sender: TObject);
20879:     procedure XMLSyntax1Click(Sender: TObject);
20880:     procedure ComponentCount1Click(Sender: TObject);
20881:     procedure NewInstance1Click(Sender: TObject);
20882:     procedure CSyntax1Click(Sender: TObject);
20883:     procedure Tutorial6Click(Sender: TObject);
20884:     procedure New1Click(Sender: TObject);
20885:     procedure AllObjectsList1Click(Sender: TObject);
20886:     procedure LoadBytecode1Click(Sender: TObject);
20887:     procedure CipherFile1Click(Sender: TObject);  //V3.5
20888:     procedure NewInstance1Click(Sender: TObject);
20889:     procedure toolbtnTutorialClick(Sender: TObject);
20890:     procedure Memory1Click(Sender: TObject);
20891:     procedure JavaSyntax1Click(Sender: TObject);
20892:     procedure SyntaxCheck1Click(Sender: TObject);
20893:     procedure ScriptExplorer1Click(Sender: TObject);
20894:     procedure FormOutput1Click(Sender: TObject);  //V3.6
20895:     procedure GotoEnd1Click(Sender: TObject);
20896:     procedure AllResourceList1Click(Sender: TObject);
20897:     procedure tbtn6resClick(Sender: TObject);   //V3.7
20898:     procedure Info1Click(Sender: TObject);
20899:     procedure Tutorial10Statistics1Click(Sender: TObject);
20900:     procedure Tutorial11Forms1Click(Sender: TObject);
20901:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
20902:     procedure ResourceExplore1Click(Sender: TObject);
20903:     procedure Info1Click(Sender: TObject);
20904:     procedure CryptoBox1Click(Sender: TObject);
20905:     procedure ModulesCount1Click(Sender: TObject);
20906:     procedure N4GewinntGame1Click(Sender: TObject);
20907:     procedure PHPSyntax1Click(Sender: TObject);
20908:     procedure SerialRS2321Click(Sender: TObject);
20909:     procedure CSyntax2Click(Sender: TObject);
20910:     procedure Calculator1Click(Sender: TObject);
20911:     procedure Tutorial13Ciphering1Click(Sender: TObject);
20912:     procedure Tutorial14Async1Click(Sender: TObject);
20913:     procedure PHPSyntax1Click(Sender: TObject);
20914:     procedure BtnZoomPlusClick(Sender: TObject);
20915:     procedure BtnZoomMinusClick(Sender: TObject);
20916:     procedure btnClassReportClick(Sender: TObject);
20917:     procedure ThreadDemo1Click(Sender: TObject);
20918:     procedure HEXView1Click(Sender: TObject);
20919:     procedure ExporttoHTML1Click(Sender: TObject);
20920:     procedure Minesweeper1Click(Sender: TObject);
20921:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
20922:     procedure sbvclhelpClick(Sender: TObject);
20923:     procedure DependencyWalker1Click(Sender: TObject);
20924:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
20925:     procedure WebScanner1Click(Sender: TObject);
20926:     procedure mnToolbar1Click(Sender: TObject);
20927:     procedure mnStatusbar2Click(Sender: TObject);
20928:     procedure mnConsole2Click(Sender: TObject);
20929:     procedure mnCoolbar2Click(Sender: TObject);
20930:     procedure mnSplitter2Click(Sender: TObject);
20931:     procedure WebServer1Click(Sender: TObject);
20932:     procedure PerlSyntax1Click(Sender: TObject);
20933:     procedure PythonSyntax1Click(Sender: TObject);
20934:     procedure DMathLibrary1Click(Sender: TObject);
20935:     procedure IntfNavigator1Click(Sender: TObject);
20936:     procedure FullTextFinder1Click(Sender: TObject);
20937:     function AppName: string;
20938:     function ScriptName: string;
20939:     function LastName: string;
20940:     procedure FractalDemo1Click(Sender: TObject);
20941:     procedure SimuLogBox1Click(Sender: TObject);
20942:     procedure OpenExamples1Click(Sender: TObject);
20943:     procedure Halt1Click(Sender: TObject);
20944:     procedure Stop;
20945:     procedure CodeSearch1Click(Sender: TObject);
20946:     procedure RubySyntax1Click(Sender: TObject);
20947:     procedure Undo1Click(Sender: TObject);
20948:     procedure LinuxShellScript1Click(Sender: TObject);
20949:     procedure WebScannerDirect(urls: string);
20950:     procedure WebScanner(urls: string);
20951:     procedure LoadInterfaceList2;
20952:     procedure DLLSpy1Click(Sender: TObject);
20953:     procedure Memo1DblClick(Sender: TObject);
20954:     procedure URILinksClicks1Click(Sender: TObject);
20955:     procedure GotoLine1Click(Sender: TObject);
20956:     procedure ConfigFile1Click(Sender: TObject);
20957:     Procedure Sort1IntflistClick( Sender : TObject)
20958:     Procedure Redo1Click( Sender : TObject)
20959:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
20960:     Procedure IndentSelection1Click( Sender : TObject)
20961:     Procedure UnindentSection1Click( Sender : TObject)
20962:     Procedure SkyStyle1Click( Sender : TObject)
20963:     Procedure CountWords1Click( Sender : TObject)
```

```
20964:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
20965:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
20966:     Procedure Bookmark11Click( Sender : TObject)
20967:     Procedure Bookmark21Click( Sender : TObject)
20968:     Procedure Bookmark31Click( Sender : TObject)
20969:     Procedure Bookmark41Click( Sender : TObject)
20970:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
20971:     'STATMemoryReport', 'boolean', iptrw);
20972:     'IPPort', 'integer', iptrw);
20973:     'COMPort', 'integer', iptrw);
20974:     'lbintflist', 'TListBox', iptrw);
20975:     Function GetStatChange : boolean
20976:     Procedure SetStatChange( vstat : boolean)
20977:     Function GetActFileName : string
20978:     Procedure SetActFileName( vname : string)
20979:     Function GetLastFileName : string
20980:     Procedure SetLastFileName( vname : string)
20981:     Procedure WebScannerDirect( urls : string)
20982:     Procedure LoadInterfaceList2
20983:     Function GetStatExecuteShell : boolean
20984:     Procedure DoEditorExecuteCommand( EditorCommand : word)
20985:     function GetActiveLineColor: TColor
20986:     procedure SetActiveLineColor(acolor: TColor)
20987:     procedure ScriptListbox1Click(Sender: TObject);
20988:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
20989:     procedure EnlargeGutter1Click(Sender: TObject);
20990:     procedure Tetris1Click(Sender: TObject);
20991:     procedure ToDoList1Click(Sender: TObject);
20992:     procedure ProcessList1Click(Sender: TObject);
20993:     procedure MetricReport1Click(Sender: TObject);
20994:     procedure ProcessList1Click(Sender: TObject);
20995:     procedure TCPSockets1Click(Sender: TObject);
20996:     procedure ConfigUpdate1Click(Sender: TObject);
20997:     procedure ADOWorkbench1Click(Sender: TObject);
20998:     procedure SocketServer1Click(Sender: TObject);
20999:     procedure FormDemo1Click(Sender: TObject);
21000:     procedure Richedit1Click(Sender: TObject);
21001:     procedure SimpleBrowser1Click(Sender: TObject);
21002:     procedure DOSShell1Click(Sender: TObject);
21003:     procedure SynExport1Click(Sender: TObject);
21004:     procedure ExporttoRTF1Click(Sender: TObject);
21005:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
21006:     procedure SOAPTester1Click(Sender: TObject);
21007:     procedure Sniffer1Click(Sender: TObject);
21008:     procedure AutoDetectSyntax1Click(Sender: TObject);
21009:     procedure FPlot1Click(Sender: TObject);
21010:     procedure PasStyle1Click(Sender: TObject);
21011:     procedure Tutorial183RGBLED1Click(Sender: TObject);
21012:     procedure Reversi1Click(Sender: TObject);
21013:     procedure ManualmaXbox1Click(Sender: TObject);
21014:     procedure BlaisePascalMagazine1Click(Sender: TObject);
21015:     procedure AddToDo1Click(Sender: TObject);
21016:     procedure CreateGUID1Click(Sender: TObject);
21017:     procedure Tutorial27XML1Click(Sender: TObject);
21018:     procedure CreateDLLStub1Click(Sender: TObject);
21019:     procedure Tutorial28DLL1Click(Sender: TObject);');
21020:     procedure ResetKeyPressed;');
21021:     procedure KeyPressedFalse;
21022:     procedure FileChanges1Click(Sender: TObject);');
21023:     procedure OpenGLTry1Click(Sender: TObject);');
21024:     procedure AllUnitList1Click(Sender: TObject);');
21025:     procedure Tutorial29UMLClick(Sender: TObject);
21026:     procedure CreateHeader1Click(Sender: TObject);
21027:     procedure Oscilloscope1Click(Sender: TObject);');
21028:     procedure Tutorial30WOT1Click(Sender: TObject);');
21029:     procedure GetWebScript1Click(Sender: TObject);');
21030:     procedure Checkers1Click(Sender: TObject);');
21031:     procedure TaskMgr1Click(Sender: TObject);');
21032:     procedure WebCam1Click(Sender: TObject);');
21033:     procedure Tutorial31Closure1Click(Sender: TObject);');
21034:     procedure GEOMapView1Click(Sender: TObject);');
21035:     procedure Run1Click(Sender: TObject);
21036:     MaxForm1.GPSSatView1Click, 'GPSSatView1Click');
21037:     MaxForm1.N3DLab1Click, 'N3DLab1Click');
21038:
21039: //-----------------------------------------------------------------------
21040: //**************mX4 Editor SynEdit Tools API ********************************
21041: //-----------------------------------------------------------------------------
21042: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
21043: begin
21044:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
21045:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
21046:     Constructor Create( AOwner : TComponent)
21047:     SelStart', 'Integer', iptrw);
21048:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
21049:     Procedure UpdateCaret
21050:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
21051:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
21052:     Procedure BeginUndoBlock
```

```
21053:     Procedure BeginUpdate
21054:     Function CaretInView : Boolean
21055:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
21056:     Procedure Clear
21057:     Procedure ClearAll
21058:     Procedure ClearBookMark( BookMark : Integer)
21059:     Procedure ClearSelection
21060:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
21061:     Procedure ClearUndo
21062:     Procedure CopyToClipboard
21063:     Procedure CutToClipboard
21064:     Procedure DoCopyToClipboard( const SText : string)
21065:     Procedure EndUndoBlock
21066:     Procedure EndUpdate
21067:     Procedure EnsureCursorPosVisible
21068:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
21069:     Procedure FindMatchingBracket
21070:     Function GetMatchingBracket : TBufferCoord
21071:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
21072:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
21073:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
21074:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
21075:               : TSynHighlighterAttributes) : boolean
21076:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
21077:               var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
21078:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
21079:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
21080:     Procedure GotoBookMark( BookMark : Integer)
21081:     Procedure GotoLineAndCenter( ALine : Integer)
21082:     Function IdentChars : TSynIdentChars
21083:     Procedure InvalidateGutter
21084:     Procedure InvalidateGutterLine( aLine : integer)
21085:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
21086:     Procedure InvalidateLine( Line : integer)
21087:     Procedure InvalidateLines( FirstLine, LastLine : integer)
21088:     Procedure InvalidateSelection
21089:     Function IsBookmark( BookMark : integer) : boolean
21090:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
21091:     Procedure LockUndo
21092:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
21093:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
21094:     Function LineToRow( aLine : integer) : integer
21095:     Function RowToLine( aRow : integer) : integer
21096:     Function NextWordPos : TBufferCoord
21097:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
21098:     Procedure PasteFromClipboard
21099:     Function WordStart : TBufferCoord
21100:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
21101:     Function WordEnd : TBufferCoord
21102:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
21103:     Function PrevWordPos : TBufferCoord
21104:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
21105:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
21106:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
21107:     Procedure Redo
21108:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
21109:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
21110:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
21111:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
21112:     Procedure SelectAll
21113:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
21114:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
21115:     Procedure SetDefaultKeystrokes
21116:     Procedure SetSelWord
21117:     Procedure SetWordBlock( Value : TBufferCoord)
21118:     Procedure Undo
21119:     Procedure UnlockUndo
21120:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
21121:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
21122:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
21123:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
21124:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
21125:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
21126:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
21127:     Procedure AddFocusControl( aControl : TWinControl)
21128:     Procedure RemoveFocusControl( aControl : TWinControl)
21129:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
21130:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
21131:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
21132:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
21133:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
21134:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
21135:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
21136:     Procedure RemoveLinesPointer
21137:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
21138:     Procedure UnHookTextBuffer
21139:     BlockBegin', 'TBufferCoord', iptrw);
21140:     BlockEnd', 'TBufferCoord', iptrw);
21141:     CanPaste', 'Boolean', iptr);
```

```
21142:     CanRedo', 'boolean', iptr);
21143:     CanUndo', 'boolean', iptr);
21144:     CaretX', 'Integer', iptrw);
21145:     CaretY', 'Integer', iptrw);
21146:     CaretXY', 'TBufferCoord', iptrw);
21147:     ActiveLineColor', 'TColor', iptrw);
21148:     DisplayX', 'Integer', iptr);
21149:     DisplayY', 'Integer', iptr);
21150:     DisplayXY', 'TDisplayCoord', iptr);
21151:     DisplayLineCount', 'integer', iptr);
21152:     CharsInWindow', 'Integer', iptr);
21153:     CharWidth', 'integer', iptr);
21154:     Font', 'TFont', iptrw);
21155:     GutterWidth', 'Integer', iptr);
21156:     Highlighter', 'TSynCustomHighlighter', iptrw);
21157:     LeftChar', 'Integer', iptrw);
21158:     LineHeight', 'integer', iptr);
21159:     LinesInWindow', 'Integer', iptr);
21160:     LineText', 'string', iptrw);
21161:     Lines', 'TStrings', iptrw);
21162:     Marks', 'TSynEditMarkList', iptr);
21163:     MaxScrollWidth', 'integer', iptrw);
21164:     Modified', 'Boolean', iptrw);
21165:     PaintLock', 'Integer', iptr);
21166:     ReadOnly', 'Boolean', iptrw);
21167:     SearchEngine', 'TSynEditSearchCustom', iptrw);
21168:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
21169:     SelTabBlock', 'Boolean', iptr);
21170:     SelTabLine', 'Boolean', iptr);
21171:     SelText', 'string', iptrw);
21172:     StateFlags', 'TSynStateFlags', iptr);
21173:     Text', 'string', iptrw);
21174:     TopLine', 'Integer', iptrw);
21175:     WordAtCursor', 'string', iptr);
21176:     WordAtMouse', 'string', iptr);
21177:     UndoList', 'TSynEditUndoList', iptr);
21178:     RedoList', 'TSynEditUndoList', iptr);
21179:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
21180:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
21181:     BorderStyle', 'TSynBorderStyle', iptrw);
21182:     ExtraLineSpacing', 'integer', iptrw);
21183:     Gutter', 'TSynGutter', iptrw);
21184:     HideSelection', 'boolean', iptrw);
21185:     InsertCaret', 'TSynEditCaretType', iptrw);
21186:     InsertMode', 'boolean', iptrw);
21187:     IsScrolling', 'Boolean', iptr);
21188:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
21189:     MaxUndo', 'Integer', iptrw);
21190:     Options', 'TSynEditorOptions', iptrw);
21191:     OverwriteCaret', 'TSynEditCaretType', iptrw);
21192:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
21193:     ScrollHintColor', 'TColor', iptrw);
21194:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
21195:     ScrollBars', 'TScrollStyle', iptrw);
21196:     SelectedColor', 'TSynSelectedColor', iptrw);
21197:     SelectionMode', 'TSynSelectionMode', iptrw);
21198:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
21199:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
21200:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
21201:     WordWrapGlyph', 'TSynGlyph', iptrw);
21202:     OnChange', 'TNotifyEvent', iptrw);
21203:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
21204:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
21205:     OnContextHelp', 'TContextHelpEvent', iptrw);
21206:     OnDropFiles', 'TDropFilesEvent', iptrw);
21207:     OnGutterClick', 'TGutterClickEvent', iptrw);
21208:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
21209:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
21210:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
21211:     OnPaint', 'TPaintEvent', iptrw);
21212:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
21213:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
21214:     OnReplaceText', 'TReplaceTextEvent', iptrw);
21215:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
21216:     OnStatusChange', 'TStatusChangeEvent', iptrw);
21217:     OnPaintTransient', 'TPaintTransient', iptrw);
21218:     OnScroll', 'TScrollEvent', iptrw);
21219:   end;
21220:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
21221:  Function GetPlaceableHighlighters : TSynHighlighterList
21222:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
21223:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
21224:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
21225:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
21226:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
21227:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
21228:  Function ConvertCodeStringToExtended( AString : String) : String
21229:  Function ConvertExtendedToCodeString( AString : String) : String
21230:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
```

```
21231:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
21232:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
21233:
21234:   TSynEditorOption = (
21235:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
21236:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
21237:                                // preceding line
21238:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
21239:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
21240:                                //direction any more
21241:     eoDragDropEditing,         //Allows to select a textblock and drag it in document to another location
21242:     eoDropFiles,               //Allows the editor accept OLE file drops
21243:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
21244:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
21245:     eoGroupUndo,               //When undoing/redoing actions,handle all cont.changes same kind in onecall
21246:                                //instead undoing/redoing each command separately
21247:     eoHalfPageScroll,          //By scrolling with page-up/page-down commands,only scroll half page attime
21248:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
21249:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
21250:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
21251:     eoNoCaret,                 //Makes it so the caret is never visible
21252:     eoNoSelection,             //Disables selecting text
21253:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
21254:     eoScrollByOneLess,         //Forces scrolling to be one less
21255:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
21256:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
21257:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
21258:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
21259:     eoShowSpecialChars,        //Shows the special Characters
21260:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
21261:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
21262:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
21263:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
21264:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
21265:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
21266:
21267:     ********************************Important Editor Short Cuts*****************************);
21268:     Double click to select a word and count words with highlightning.
21269:     Triple click to select a line.
21270:     CTRL+SHIFT+click to extend a selection.
21271:     Drag with the ALT key down to select columns of text !!!
21272:     Drag and drop is supported.
21273:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
21274:     Type CTRL+A to select all.
21275:     Type CTRL+N to set a new line.
21276:     Type CTRL+T to delete a line or token. //Tokenizer
21277:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
21278:     Type CTRL+Shift+T to add ToDo in line and list.
21279:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
21280:     Type CTRL[0..9] to jump or get to bookmarks.
21281:     Type Home to position cursor at beginning of current line and End to position it at end of line.
21282:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
21283:     Page Up and Page Down work as expected.
21284:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
21285:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
21286:
21287: {$ Short Key Positions Ctrl<A-Z>: }
21288: def
21289:    <A> Select All
21290:    <B> Count Words
21291:    <C> Copy
21292:    <D> Internet Start
21293:    <E> Script List
21294:    <F> Find
21295:    <G> Goto
21296:    <H> Mark Line
21297:    <I> Interface List
21298:    <J> Code Completion
21299:    <K> Console
21300:    <L> Interface List Box
21301:    <M> Font Larger -
21302:    <N> New Line
21303:    <O> Open File
21304:    <P> Font Smaller +
21305:    <Q> Quit
21306:    <R> Replace
21307:    <S> Save!
21308:    <T> Delete Line
21309:    <U> Use Case Editor
21310:    <V> Paste
21311:    <W> URI Links
21312:    <X> Reserved for coding use internal
21313:    <Y> Delete Line
21314:    <Z> Undo
21315:
21316: ref
21317:    F1 Help
21318:    F2 Syntax Check
21319:    F3 Search Next
```

```
21320:    F4 New Instance
21321:    F5 Line Mark /Breakpoint
21322:    F6 Goto End
21323:    F7 Debug Step Into
21324:    F8 Debug Step Out
21325:    F9 Compile
21326:    F10 Menu
21327:    F11 Word Count Highlight
21328:    F12 Reserved for coding use internal
21329:
21330:    AddRegisteredVariable('Application', 'TApplication');
21331:    AddRegisteredVariable('Screen', 'TScreen');
21332:    AddRegisteredVariable('Self', 'TForm');
21333:    AddRegisteredVariable('Memo1', 'TSynMemo');
21334:    AddRegisteredVariable('memo2', 'TMemo');
21335:    AddRegisteredVariable('maxForm1', 'TMaxform1');  //!!
21336:    AddRegisteredVariable('debugout', 'Tdebugoutput');  //!!
21337:    AddRegisteredVariable('hlog','THotlog');  //!!
21338:    AddRegisteredVariable( it ,integer');  //for closure!!
21339:    AddRegisteredVariable( sr ,string');  //for closure
21340:    AddRegisteredVariable( bt ,boolean');  //for closure
21341:    AddRegisteredVariable( ft ,double');  //for closure
21342:    AddRegisteredVariable( srlist ,TStringlist');  //for closures
21343:
21344:  def ReservedWords: array[0..82] of string =
21345:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
21346:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
21347:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
21348:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
21349:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
21350:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
21351:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
21352:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
21353:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
21354:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
21355:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
21356:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
21357: //-----------------------------------------------------------------------------
21358: //*************End of mX4 Public  Tools API *********************************
21359: //-----------------------------------------------------------------------------
21360:
21361: Amount of Functions: 13685
21362: Amount of Procedures: 8396
21363: Amount of Constructors: 1369
21364: Totals of Calls: 23450
21365: SHA1: Win 3.9.9.101 786FA3035226AC3F62F6F6B9E2E2B94C1D659297
21366:
21367: ************************************************************
21368:  Doc Short Manual with 50 Tips!
21369: ************************************************************
21370: - Install: just save your maxboxdef.ini before and then extract the zip file!
21371: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
21372: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
21373: - Menu: With <Crtl><F3> you can search for code on examples
21374: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
21375: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
21376: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
21377:
21378: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
21379: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
21380: - Context Menu: You can printout your scripts as a pdf-file or html-export
21381: - Context: You do have a context menu with the right mouse click
21382:
21383: - Menu: With the UseCase Editor you can convert graphic formats too.
21384: - Menu: On menu Options you find Addons as compiled scripts
21385: - IDE: Menu Program: Run Only is faster, after F2 - You don't need a mouse use shortcuts
21386: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
21387: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
21388: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
21389:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
21390:
21391: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
21392: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
21393: - Code: If you code a loop till key-pressed use function: isKeyPressed;
21394: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
21395: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
21396: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
21397:          to delete and Click and mark to drag a bookmark
21398: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
21399: - IDE: A file info with system and script information you find in menu Program/Information
21400: - IDE: After change the config file in help you can update changes in menu Help/Config Update
21401: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
21402: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
21403: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
21404: - Editor: Set Bookmarks to check your work in app or code
21405: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
21406: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
21407: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
21408:
```

```
21409: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
21410: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
21411: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
21412: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
21413: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
21414: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
21415: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
21416: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
21417: - IDE menu /Help/Tools/ open the Task Manager
21418:
21419: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
21420: - Add on when no browser is available start /Options/Add ons/Easy Browser
21421: - Add on SOAP Tester with SOP POST File
21422: - Add on IP Protocol Sniffer with List View
21423: - Add on OpenGL mX Robot Demo for android
21424: - Add on Checkers Game, Add on Oscilloscope
21425:
21426: - Menu: Help/Tools as a Tool Section with DOS Opener
21427: - Menu Editor: export the code as RTF File
21428: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
21429: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
21430: - Context: Auto Detect of Syntax depending on file extension
21431: - Code: some Windows API function start with w in the name like wGetAtomName();
21432: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
21433: - IDE File Check with menu ..View/File Changes/...
21434: - Context: Create a Header with Create Header in Navigator List at right window
21435: - Code: use  SysErrorMessage to get a real Error Description, Ex.
21436:         RemoveDir('c:\NoSuchFolder') ;
21437:         writeln('System Error Message: '+ SysErrorMessage(GetLastError));
21438: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
21439:
21440: - using DLL example in maXbox: //function: {*****************************************}
21441:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
21442:                                  cb: DWORD): BOOL; //stdcall;;
21443:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
21444:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
21445:      External  'OpenProcess@kernel32.dll stdcall';
21446:
21447: GCC Compile Ex Script
21448: procedure TFormMain_btnCompileClick(Sender: TObject);
21449: begin
21450:   AProcess:= TProcess.Create(Nil);
21451:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
21452:       + ' -o "' + OpenDialog2.FileName + '"';
21453:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
21454:     AProcess.Execute;
21455:     Memo2.Lines.BeginUpdate;
21456:     Memo2.Lines.Clear;
21457:     Memo2.Lines.LoadFromStream(AProcess.Output);
21458:     Memo2.Lines.EndUpdate;
21459:   finally
21460:     AProcess.Free;
21461:   end;
21462: end;
21463:
21464: Stopwatch pattern
21465: Time1:= Time;
21466:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
21467:   if initAndStartBoard then
21468:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
21469:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
21470:   PrintF('%d %s',[Trunc((Time-Time1)*24),
21471:                   FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
21472:
21473:    POST git-receive-pack (chunked)
21474: Pushing to https://github.com/maxkleiner/maXbox3.git
21475: To https://github.com/maxkleiner/maXbox3.git
21476:    f127d21..c6a98da  masterbox2 -> masterbox2
21477: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
21478:
21479: History Shell Hell
21480: PCT Precompile Technology , mX4 ScriptStudio
21481: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
21482: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
21483: emax layers: system-package-component-unit-class-function-block
21484: new keywords def ref using maXCalcF
21485: UML: use case act class state seq pac comp dep - lib lab
21486: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
21487: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
21488: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
21489: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
21490: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
21491: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
21492: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
21493: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
21494: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
21495: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
21496: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21497: Inno Install and Setup Routines
```

```
21498: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21499: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21500: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21501: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
21502: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
21503: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21504: StreamUtils, IDL Syntax, OpenStreetMap, runByteCode, sensor panel, CGI of Powtils
21505: ByteCode2, IPUtils2, GEOCode
21506:
21507: Ref:
21508: https://unibe-ch.academia.edu/MaxKleiner
21509: http://www.slideshare.net/maxkleiner1
21510: http://www.scribd.com/max_kleiner
21511: http://www.delphiforfun.org/Programs/Utilities/index.htm
21512: http://www.slideshare.net/maxkleiner1
21513: http://s3.amazonaws.com/PreviewLinks/22959.html
21514: http://www.softwareschule.ch/arduino_training.pdf
21515: http://www.jrsoftware.org/isinfo.php
21516: http://www.be-precision.com/products/precision-builder/express/
21517: http://www.blaisepascal.eu/
21518: http://www.delphibasics.co.uk/
21519: http://www.youtube.com/watch?v=av89HAbqAsI
21520: http://www.angelfire.com/hi5/delphizeus/modal.html
21521: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
21522: http://delphi.org/2014/01/every-android-api-for-delphi/
21523: https://en.wikipedia.org/wiki/User:Maxkleiner
21524: http://en.wikipedia.org/wiki/Megido_%28Free_Pascal%29
21525:
21526:
21527: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
21528: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
21529: =renderBasicSearchNarrative&q=%s';
21530: UrlMapQuestAPIReverse:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
21531: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
21532:
21533:
21534: function OpenMap(const Data: string): boolean;
21535: var encURL: string;
21536: begin
21537:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
21538:   try
21539:    //HttpGet(EncodedURL, mapStream);   //WinInet
21540:    Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
21541:    //OpenDoc(Exepath+'openmapx.html');
21542:    S_ShellExecute(Exepath+'openmapx.html','',seCmdOpen);
21543:   finally
21544:     encURL:= '';
21545:   end;
21546: end;
21547:
21548: procedure GetGEOMap(C_form,apath: string; const Data: string);
21549: var
21550:   encodedURL: string;
21551:   mapStream: TMemoryStream;
21552: begin
21553:   //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPEncode(Data)]);
21554:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
21555:   mapStream:= TMemoryStream.create;
21556:   try
21557:     Wininet_HttpGet(EncodedURL, mapStream);   //WinInet
21558:     mapStream.Position:= 0;
21559:     mapStream.Savetofile(apath);
21560:     // OpenDoc(apath);
21561:     S_ShellExecute(apath,'',seCmdOpen);
21562:   finally
21563:     mapStream.Free;
21564:   end;
21565: end;
21566:
21567:
21568: ************************************************************
21569:  unit List asm internal end
21570: ************************************************************
21571: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
21572: 02 unit SIRegister_IdStrings                   //Indy Sockets
21573: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
21574: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
21575: 05 unit IFSI_WinForm1puzzle;                   //maXbox
21576: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
21577: 07 unit RegisterDateTimeLibrary_R(exec);       //Delphi
21578: 08 unit RIRegister_MathMax_Routines(exec);     //Jedi & Delphi
21579: 09 unit RIRegister_IdGlobal_Routines(exec);    //Indy Sockets
21580: 10 unit RIRegister_SysUtils_Routines(Exec);    //Delphi
21581: 11 unit uPSI_IdTCPConnection;                  //Indy some functions
21582: 12 unit uPSCompiler.pas;                       //PS kernel functions
21583: 13 unit uPSI_DBCommon;                         //DB Common_Routines and Types
21584: 14 unit uPSI_Printers.pas                      //Delphi VCL
21585: 15 unit uPSI_MPlayer.pas                       //Delphi VCL
21586: 16 unit uPSC_comobj;                          //COM Functions
```

```
21587: 17 unit uPSI_Clipbrd;                        //Delphi VCL
21588: 18 unit Filectrl in IFSI_SysUtils_max;       //VCL Runtime
21589: 19 unit uPSI_SqlExpr;                        //DBX3
21590: 20 unit uPSI_ADODB;                          //ADODB
21591: 21 unit uPSI_StrHlpr;                        //String Helper Routines
21592: 22 unit uPSI_DateUtils;                      //Expansion to DateTimeLib
21593: 23 unit uPSI_FileUtils;                      //Expansion to Sys/File Utils
21594: 24 unit JUtils / gsUtils;                    //Jedi / Metabase
21595: 25 unit JvFunctions_max;                     //Jedi Functions
21596: 26 unit HTTPParser;                          //Delphi VCL
21597: 27 unit HTTPUtil;                            //Delphi VCL
21598: 28 unit uPSI_XMLUtil;                       //Delphi VCL
21599: 29 unit uPSI_SOAPHTTPClient;                 //Delphi VCL SOAP WebService V3.5
21600: 30 unit uPSI_Contnrs;                       //Delphi RTL Container of Classes
21601: 31 unit uPSI_MaskUtils;                     //RTL Edit and Mask functions
21602: 32 unit uPSI_MyBigInt;                      //big integer class with Math
21603: 33 unit uPSI_ConvUtils;                     //Delphi VCL Conversions engine
21604: 34 unit Types_Variants;                     //Delphi\Win32\rtl\sys
21605: 35 unit uPSI_IdHashSHA1;                    //Indy Crypto Lib
21606: 36 unit uPSI_IdHashMessageDigest           //Indy Crypto;
21607: 37 unit uPSI_IdASN1Util;                    //Indy ASN1Utility Routines;
21608: 38 unit uPSI_IdLogFile;                     //Indy Logger from LogBase
21609: 39 unit uPSI_IdIcmpClient;                  //Indy Ping ICMP
21610: 40 unit uPSI_IdHashMessageDigest_max       //Indy Crypto &OpenSSL;
21611: 41 unit uPSI_FileCtrl;                      //Delphi RTL
21612: 42 unit uPSI_Outline;                       //Delphi VCL
21613: 43 unit uPSI_ScktComp;                      //Delphi RTL
21614: 44 unit uPSI_Calendar;                      //Delphi VCL
21615: 45 unit uPSI_VListView                      //VListView;
21616: 46 unit uPSI_DBGrids;                       //Delphi VCL
21617: 47 unit uPSI_DBCtrls;                       //Delphi VCL
21618: 48 unit ide_debugoutput;                    //maXbox
21619: 49 unit uPSI_ComCtrls;                     //Delphi VCL
21620: 50 unit uPSC_stdctrls+;                    //Delphi VCL
21621: 51 unit uPSI_Dialogs;                      //Delphi VCL
21622: 52 unit uPSI_StdConvs;                     //Delphi RTL
21623: 53 unit uPSI_DBClient;                     //Delphi RTL
21624: 54 unit uPSI_DBPlatform;                   //Delphi RTL
21625: 55 unit uPSI_Provider;                     //Delphi RTL
21626: 56 unit uPSI_FMTBcd;                       //Delphi RTL
21627: 57 unit uPSI_DBCGrids;                     //Delphi VCL
21628: 58 unit uPSI_CDSUtil;                      //MIDAS
21629: 59 unit uPSI_VarHlpr;                      //Delphi RTL
21630: 60 unit uPSI_ExtDlgs;                      //Delphi VCL
21631: 61 unit sdpStopwatch;                      //maXbox
21632: 62 unit uPSI_JclStatistics;                //JCL
21633: 63 unit uPSI_JclLogic;                     //JCL
21634: 64 unit uPSI_JclMiscel;                    //JCL
21635: 65 unit uPSI_JclMath_max;                  //JCL RTL
21636: 66 unit uPSI_uTPLb_StreamUtils;            //LockBox 3
21637: 67 unit uPSI_MathUtils;                    //BCB
21638: 68 unit uPSI_JclMultimedia;                //JCL
21639: 69 unit uPSI_WideStrUtils;                 //Delphi API/RTL
21640: 70 unit uPSI_GraphUtil;                    //Delphi RTL
21641: 71 unit uPSI_TypeTrans;                    //Delphi RTL
21642: 72 unit uPSI_HTTPApp;                      //Delphi VCL
21643: 73 unit uPSI_DBWeb;                        //Delphi VCL
21644: 74 unit uPSI_DBBdeWeb;                     //Delphi VCL
21645: 75 unit uPSI_DBXpressWeb;                  //Delphi VCL
21646: 76 unit uPSI_ShadowWnd;                    //Delphi VCL
21647: 77 unit uPSI_ToolWin;                      //Delphi VCL
21648: 78 unit uPSI_Tabs;                         //Delphi VCL
21649: 79 unit uPSI_JclGraphUtils;                //JCL
21650: 80 unit uPSI_JclCounter;                   //JCL
21651: 81 unit uPSI_JclSysInfo;                   //JCL
21652: 82 unit uPSI_JclSecurity;                  //JCL
21653: 83 unit uPSI_JclFileUtils;                 //JCL
21654: 84 unit uPSI_IdUserAccounts;               //Indy
21655: 85 unit uPSI_IdAuthentication;             //Indy
21656: 86 unit uPSI_uTPLb_AES;                    //LockBox 3
21657: 87 unit uPSI_IdHashSHA1;                   //LockBox 3
21658: 88 unit uTPLb_BlockCipher;                 //LockBox 3
21659: 89 unit uPSI_ValEdit.pas;                  //Delphi VCL
21660: 90 unit uPSI_JvVCLUtils;                   //JCL
21661: 91 unit uPSI_JvDBUtil;                     //JCL
21662: 92 unit uPSI_JvDBUtils;                    //JCL
21663: 93 unit uPSI_JvAppUtils;                   //JCL
21664: 94 unit uPSI_JvCtrlUtils;                  //JCL
21665: 95 unit uPSI_JvFormToHtml;                 //JCL
21666: 96 unit uPSI_JvParsing;                    //JCL
21667: 97 unit uPSI_SerDlgs;                      //Toolbox
21668: 98 unit uPSI_Serial;                       //Toolbox
21669: 99 unit uPSI_JvComponent;                  //JCL
21670: 100 unit uPSI_JvCalc;                      //JCL
21671: 101 unit uPSI_JvBdeUtils;                  //JCL
21672: 102 unit uPSI_JvDateUtil;                  //JCL
21673: 103 unit uPSI_JvGenetic;                   //JCL
21674: 104 unit uPSI_JclBase;                     //JCL
21675: 105 unit uPSI_JvUtils;                     //JCL
```

```
21676: 106 unit uPSI_JvStrUtil;                    //JCL
21677: 107 unit uPSI_JvStrUtils;                   //JCL
21678: 108 unit uPSI_JvFileUtil;                   //JCL
21679: 109 unit uPSI_JvMemoryInfos;                //JCL
21680: 110 unit uPSI_JvComputerInfo;               //JCL
21681: 111 unit uPSI_JvgCommClasses;               //JCL
21682: 112 unit uPSI_JvgLogics;                    //JCL
21683: 113 unit uPSI_JvLED;                        //JCL
21684: 114 unit uPSI_JvTurtle;                     //JCL
21685: 115 unit uPSI_SortThds; unit uPSI_ThSort;   //maXbox
21686: 116 unit uPSI_JvgUtils;                    //JCL
21687: 117 unit uPSI_JvExprParser;                //JCL
21688: 118 unit uPSI_HexDump;                     //Borland
21689: 119 unit uPSI_DBLogDlg;                    //VCL
21690: 120 unit uPSI_SqlTimSt;                    //RTL
21691: 121 unit uPSI_JvHtmlParser;                //JCL
21692: 122 unit uPSI_JvgXMLSerializer;            //JCL
21693: 123 unit uPSI_JvJCLUtils;                  //JCL
21694: 124 unit uPSI_JvStrings;                   //JCL
21695: 125 unit uPSI_uTPLb_IntegerUtils;          //TurboPower
21696: 126 unit uPSI_uTPLb_HugeCardinal;          //TurboPower
21697: 127 unit uPSI_uTPLb_HugeCardinalUtils;     //TurboPower
21698: 128 unit uPSI_SynRegExpr;                  //SynEdit
21699: 129 unit uPSI_StUtils;                     //SysTools4
21700: 130 unit uPSI_StToHTML;                    //SysTools4
21701: 131 unit uPSI_StStrms;                     //SysTools4
21702: 132 unit uPSI_StFIN;                       //SysTools4
21703: 133 unit uPSI_StAstroP;                    //SysTools4
21704: 134 unit uPSI_StStat;                      //SysTools4
21705: 135 unit uPSI_StNetCon;                    //SysTools4
21706: 136 unit uPSI_StDecMth;                    //SysTools4
21707: 137 unit uPSI_StOStr;                      //SysTools4
21708: 138 unit uPSI_StPtrns;                     //SysTools4
21709: 139 unit uPSI_StNetMsg;                    //SysTools4
21710: 140 unit uPSI_StMath;                      //SysTools4
21711: 141 unit uPSI_StExpEng;                    //SysTools4
21712: 142 unit uPSI_StCRC;                       //SysTools4
21713: 143 unit uPSI_StExport,                    //SysTools4
21714: 144 unit uPSI_StExpLog,                    //SysTools4
21715: 145 unit uPSI_ActnList;                    //Delphi VCL
21716: 146 unit uPSI_jpeg;                        //Borland
21717: 147 unit uPSI_StRandom;                    //SysTools4
21718: 148 unit uPSI_StDict;                      //SysTools4
21719: 149 unit uPSI_StBCD;                       //SysTools4
21720: 150 unit uPSI_StTxtDat;                    //SysTools4
21721: 151 unit uPSI_StRegEx;                     //SysTools4
21722: 152 unit uPSI_IMouse;                      //VCL
21723: 153 unit uPSI_SyncObjs;                    //VCL
21724: 154 unit uPSI_AsyncCalls;                  //Hausladen
21725: 155 unit uPSI_ParallelJobs;                //Saraiva
21726: 156 unit uPSI_Variants;                    //VCL
21727: 157 unit uPSI_VarCmplx;                    //VCL Wolfram
21728: 158 unit uPSI_DTDSchema;                   //VCL
21729: 159 unit uPSI_ShLwApi;                     //Brakel
21730: 160 unit uPSI_IBUtils;                     //VCL
21731: 161 unit uPSI_CheckLst;                    //VCL
21732: 162 unit uPSI_JvSimpleXml;                 //JCL
21733: 163 unit uPSI_JclSimpleXml;                //JCL
21734: 164 unit uPSI_JvXmlDatabase;               //JCL
21735: 165 unit uPSI_JvMaxPixel;                  //JCL
21736: 166 unit uPSI_JvItemsSearchs;              //JCL
21737: 167 unit uPSI_StExpEng2;                   //SysTools4
21738: 168 unit uPSI_StGenLog;                    //SysTools4
21739: 169 unit uPSI_JvLogFile;                   //Jcl
21740: 170 unit uPSI_CPort;                       //ComPort Lib v4.11
21741: 171 unit uPSI_CPortCtl;                    //ComPort
21742: 172 unit uPSI_CPortEsc;                    //ComPort
21743: 173 unit BarCodeScaner;                    //ComPort
21744: 174 unit uPSI_JvGraph;                     //JCL
21745: 175 unit uPSI_JvComCtrls;                  //JCL
21746: 176 unit uPSI_GUITesting;                  //D Unit
21747: 177 unit uPSI_JvFindFiles;                 //JCL
21748: 178 unit uPSI_StSystem;                    //SysTools4
21749: 179 unit uPSI_JvKeyboardStates;            //JCL
21750: 180 unit uPSI_JvMail;                      //JCL
21751: 181 unit uPSI_JclConsole;                  //JCL
21752: 182 unit uPSI_JclLANMan;                   //JCL
21753: 183 unit uPSI_IdCustomHTTPServer;          //Indy
21754: 184 unit IdHTTPServer                      //Indy
21755: 185 unit uPSI_IdTCPServer;                 //Indy
21756: 186 unit uPSI_IdSocketHandle;              //Indy
21757: 187 unit uPSI_IdIOHandlerSocket;           //Indy
21758: 188 unit IdIOHandler;                      //Indy
21759: 189 unit uPSI_cutils;                      //Bloodshed
21760: 190 unit uPSI_BoldUtils;                   //boldsoft
21761: 191 unit uPSI_IdSimpleServer;              //Indy
21762: 192 unit uPSI_IdSSLOpenSSL;                //Indy
21763: 193 unit uPSI_IdMultipartFormData;         //Indy
21764: 194 unit uPSI_SynURIOpener;                //SynEdit
```

```
21765: 195 unit uPSI_PerlRegEx;                    //PCRE
21766: 196 unit uPSI_IdHeaderList;                 //Indy
21767: 197 unit uPSI_StFirst;                      //SysTools4
21768: 198 unit uPSI_JvCtrls;                      //JCL
21769: 199 unit uPSI_IdTrivialFTPBase;             //Indy
21770: 200 unit uPSI_IdTrivialFTP;                 //Indy
21771: 201 unit uPSI_IdUDPBase;                    //Indy
21772: 202 unit uPSI_IdUDPClient;                  //Indy
21773: 203 unit uPSI_utypes;                       //for DMath.DLL
21774: 204 unit uPSI_ShellAPI;                     //Borland
21775: 205 unit uPSI_IdRemoteCMDClient;            //Indy
21776: 206 unit uPSI_IdRemoteCMDServer;            //Indy
21777: 207 unit IdRexecServer;                     //Indy
21778: 208 unit IdRexec; (unit uPSI_IdRexec;)      //Indy
21779: 209 unit IdUDPServer;                       //Indy
21780: 210 unit IdTimeUDPServer;                   //Indy
21781: 211 unit IdTimeServer;                      //Indy
21782: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
21783: 213 unit uPSI_IdIPWatch;                    //Indy
21784: 214 unit uPSI_IdIrcServer;                  //Indy
21785: 215 unit uPSI_IdMessageCollection;          //Indy
21786: 216 unit uPSI_cPEM;                         //Fundamentals 4
21787: 217 unit uPSI_cFundamentUtils;              //Fundamentals 4
21788: 218 unit uPSI_uwinplot;                     //DMath
21789: 219 unit uPSI_xrtl_util_CPUUtils;           //ExtentedRTL
21790: 220 unit uPSI_GR32_System;                  //Graphics32
21791: 221 unit uPSI_cFileUtils;                   //Fundamentals 4
21792: 222 unit uPSI_cDateTime; (timemachine)      //Fundamentals 4
21793: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
21794: 224 unit uPSI_cRandom;                      //Fundamentals 4
21795: 225 unit uPSI_ueval;                        //DMath
21796: 226 unit uPSI_xrtl_net_URIUtils;            //ExtendedRTL
21797: 227 unit xrtl_net_URIUtils;                 //ExtendedRTL
21798: 228 unit uPSI_ufft;  (FFT)                  //DMath
21799: 229 unit uPSI_DBXChannel;                   //Delphi
21800: 230 unit uPSI_DBXIndyChannel;               //Delphi Indy
21801: 231 unit uPSI_xrtl_util_COMCat;             //ExtendedRTL
21802: 232 unit uPSI_xrtl_util_StrUtils;           //ExtendedRTL
21803: 233 unit uPSI_xrtl_util_VariantUtils;       //ExtendedRTL
21804: 234 unit uPSI_xrtl_util_FileUtils;          //ExtendedRTL
21805: 235 unit xrtl_util_Compat;                  //ExtendedRTL
21806: 236 unit uPSI_OleAuto;                      //Borland
21807: 237 unit uPSI_xrtl_util_COMUtils;           //ExtendedRTL
21808: 238 unit uPSI_CmAdmCtl;                     //Borland
21809: 239 unit uPSI_ValEdit2;                     //VCL
21810: 240 unit uPSI_GR32;  //Graphics32          //Graphics32
21811: 241 unit uPSI_GR32_Image;                   //Graphics32
21812: 242 unit uPSI_xrtl_util_TimeUtils;          //ExtendedRTL
21813: 243 unit uPSI_xrtl_util_TimeZone;           //ExtendedRTL
21814: 244 unit uPSI_xrtl_util_TimeStamp;          //ExtendedRTL
21815: 245 unit uPSI_xrtl_util_Map;                //ExtendedRTL
21816: 246 unit uPSI_xrtl_util_Set;                //ExtendedRTL
21817: 247 unit uPSI_CPortMonitor;                 //ComPort
21818: 248 unit uPSI_StIniStm;                     //SysTools4
21819: 249 unit uPSI_GR32_ExtImage;                //Graphics32
21820: 250 unit uPSI_GR32_OrdinalMaps;             //Graphics32
21821: 251 unit uPSI_GR32_Rasterizers;             //Graphics32
21822: 252 unit uPSI_xrtl_util_Exception;          //ExtendedRTL
21823: 253 unit uPSI_xrtl_util_Value;              //ExtendedRTL
21824: 254 unit uPSI_xrtl_util_Compare;            //ExtendedRTL
21825: 255 unit uPSI_FlatSB;                       //VCL
21826: 256 unit uPSI_JvAnalogClock;                //JCL
21827: 257 unit uPSI_JvAlarms;                     //JCL
21828: 258 unit uPSI_JvSQLS;                       //JCL
21829: 259 unit uPSI_JvDBSecur;                    //JCL
21830: 260 unit uPSI_JvDBQBE;                      //JCL
21831: 261 unit uPSI_JvStarfield;                  //JCL
21832: 262 unit uPSI_JVCLMiscal;                   //JCL
21833: 263 unit uPSI_JvProfiler32;                 //JCL
21834: 264 unit uPSI_JvDirectories,                //JCL
21835: 265 unit uPSI_JclSchedule,                  //JCL
21836: 266 unit uPSI_JclSvcCtrl,                   //JCL
21837: 267 unit uPSI_JvSoundControl,               //JCL
21838: 268 unit uPSI_JvBDESQLScript,               //JCL
21839: 269 unit uPSI_JvgDigits,                    //JCL>
21840: 270 unit uPSI_ImgList;                      //TCustomImageList
21841: 271 unit uPSI_JclMIDI;                      //JCL>
21842: 272 unit uPSI_JclWinMidi;                   //JCL>
21843: 273 unit uPSI_JclNTFS;                      //JCL>
21844: 274 unit uPSI_JclAppInst;                   //JCL>
21845: 275 unit uPSI_JvRle;                        //JCL>
21846: 276 unit uPSI_JvRas32;                      //JCL>
21847: 277 unit uPSI_JvImageDrawThread,            //JCL>
21848: 278 unit uPSI_JvImageWindow,                //JCL>
21849: 279 unit uPSI_JvTransparentForm;            //JCL>
21850: 280 unit uPSI_JvWinDialogs;                 //JCL>
21851: 281 unit uPSI_JvSimLogic,                   //JCL>
21852: 282 unit uPSI_JvSimIndicator,               //JCL>
21853: 283 unit uPSI_JvSimPID,                     //JCL>
```

```
21854: 284 unit uPSI_JvSimPIDLinker,                //JCL>
21855: 285 unit uPSI_IdRFCReply;                    //Indy
21856: 286 unit uPSI_IdIdent;                       //Indy
21857: 287 unit uPSI_IdIdentServer;                 //Indy
21858: 288 unit uPSI_JvPatchFile;                   //JCL
21859: 289 unit uPSI_StNetPfm;                      //SysTools4
21860: 290 unit uPSI_StNet;                         //SysTools4
21861: 291 unit uPSI_JclPeImage;                    //JCL
21862: 292 unit uPSI_JclPrint;                      //JCL
21863: 293 unit uPSI_JclMime;                       //JCL
21864: 294 unit uPSI_JvRichEdit;                    //JCL
21865: 295 unit uPSI_JvDBRichEd;                    //JCL
21866: 296 unit uPSI_JvDice;                        //JCL
21867: 297 unit uPSI_JvFloatEdit;                   //JCL 3.9.8
21868: 298 unit uPSI_JvDirFrm;                      //JCL
21869: 299 unit uPSI_JvDualList;                    //JCL
21870: 300 unit uPSI_JvSwitch;                      ////JCL
21871: 301 unit uPSI_JvTimerLst;                    ////JCL
21872: 302 unit uPSI_JvMemTable;                    //JCL
21873: 303 unit uPSI_JvObjStr;                      //JCL
21874: 304 unit uPSI_StLArr;                        //SysTools4
21875: 305 unit uPSI_StWmDCpy;                      //SysTools4
21876: 306 unit uPSI_StText;                        //SysTools4
21877: 307 unit uPSI_StNTLog;                       //SysTools4
21878: 308 unit uPSI_xrtl_math_Integer;             //ExtendedRTL
21879: 309 unit uPSI_JvImagPrvw;                    //JCL
21880: 310 unit uPSI_JvFormPatch;                   //JCL
21881: 311 unit uPSI_JvPicClip;                     //JCL
21882: 312 unit uPSI_JvDataConv;                    //JCL
21883: 313 unit uPSI_JvCpuUsage;                    //JCL
21884: 314 unit uPSI_JclUnitConv_mX2;               //JCL
21885: 315 unit JvDualListForm;                     //JCL
21886: 316 unit uPSI_JvCpuUsage2;                   //JCL
21887: 317 unit uPSI_JvParserForm;                  //JCL
21888: 318 unit uPSI_JvJanTreeView;                 //JCL
21889: 319 unit uPSI_JvTransLED;                    //JCL
21890: 320 unit uPSI_JvPlaylist;                    //JCL
21891: 321 unit uPSI_JvFormAutoSize;                //JCL
21892: 322 unit uPSI_JvYearGridEditForm;            //JCL
21893: 323 unit uPSI_JvMarkupCommon;                //JCL
21894: 324 unit uPSI_JvChart;                       //JCL
21895: 325 unit uPSI_JvXPCore;                      //JCL
21896: 326 unit uPSI_JvXPCoreUtils;                 //JCL
21897: 327 unit uPSI_StatsClasses;                  //mX4
21898: 328 unit uPSI_ExtCtrls2;                     //VCL
21899: 329 unit uPSI_JvUrlGrabbers;                 //JCL
21900: 330 unit uPSI_JvXmlTree;                     //JCL
21901: 331 unit uPSI_JvWavePlayer;                  //JCL
21902: 332 unit uPSI_JvUnicodeCanvas;               //JCL
21903: 333 unit uPSI_JvTFUtils;                     //JCL
21904: 334 unit uPSI_IdServerIOHandler;             //Indy
21905: 335 unit uPSI_IdServerIOHandlerSocket;       //Indy
21906: 336 unit uPSI_IdMessageCoder;                //Indy
21907: 337 unit uPSI_IdMessageCoderMIME;            //Indy
21908: 338 unit uPSI_IdMIMETypes;                   //Indy
21909: 339 unit uPSI_JvConverter;                   //JCL
21910: 340 unit uPSI_JvCsvParse;                    //JCL
21911: 341 unit uPSI_umath;  unit uPSI_ugamma;      //DMath
21912: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
21913: 343 unit uPSI_JvDBGridExport;                //JCL
21914: 344 unit uPSI_JvgExport;                     //JCL
21915: 345 unit uPSI_JvSerialMaker;                 //JCL
21916: 346 unit uPSI_JvWin32;                       //JCL
21917: 347 unit uPSI_JvPaintFX;                     //JCL
21918: 348 unit uPSI_JvOracleDataSet; (beta)        //JCL
21919: 349 unit uPSI_JvValidators; (preview)        //JCL
21920: 350 unit uPSI_JvNTEventLog;                  //JCL
21921: 351 unit uPSI_ShellZipTool;                  //mX4
21922: 352 unit uPSI_JvJoystick;                    //JCL
21923: 353 unit uPSI_JvMailSlots;                   //JCL
21924: 354 unit uPSI_JclComplex;                    //JCL
21925: 355 unit uPSI_SynPdf;                        //Synopse
21926: 356 unit uPSI_Registry;                      //VCL
21927: 357 unit uPSI_TlHelp32;                      //VCL
21928: 358 unit uPSI_JclRegistry;                   //JCL
21929: 359 unit uPSI_JvAirBrush;                    //JCL
21930: 360 unit uPSI_mORMotReport;                  //Synopse
21931: 361 unit uPSI_JclLocales;                    //JCL
21932: 362 unit uPSI_SynEdit;                       //SynEdit
21933: 363 unit uPSI_SynEditTypes;                  //SynEdit
21934: 364 unit uPSI_SynMacroRecorder;              //SynEdit
21935: 365 unit uPSI_LongIntList;                   //SynEdit
21936: 366 unit uPSI_devcutils;                     //DevC
21937: 367 unit uPSI_SynEditMiscClasses;            //SynEdit
21938: 368 unit uPSI_SynEditRegexSearch;            //SynEdit
21939: 369 unit uPSI_SynEditHighlighter;            //SynEdit
21940: 370 unit uPSI_SynHighlighterPas;             //SynEdit
21941: 371 unit uPSI_JvSearchFiles;                 //JCL
21942: 372 unit uPSI_SynHighlighterAny;             //Lazarus
```

```
21943: 373 unit uPSI_SynEditKeyCmds;                //SynEdit
21944: 374 unit uPSI_SynEditMiscProcs,              //SynEdit
21945: 375 unit uPSI_SynEditKbdHandler              //SynEdit
21946: 376 unit uPSI_JvAppInst,                     //JCL
21947: 377 unit uPSI_JvAppEvent;                    //JCL
21948: 378 unit uPSI_JvAppCommand;                  //JCL
21949: 379 unit uPSI_JvAnimTitle;                   //JCL
21950: 380 unit uPSI_JvAnimatedImage;               //JCL
21951: 381 unit uPSI_SynEditExport;                 //SynEdit
21952: 382 unit uPSI_SynExportHTML;                 //SynEdit
21953: 383 unit uPSI_SynExportRTF;                  //SynEdit
21954: 384 unit uPSI_SynEditSearch;                 //SynEdit
21955: 385 unit uPSI_fMain_back                     //maXbox;
21956: 386 unit uPSI_JvZoom;                        //JCL
21957: 387 unit uPSI_PMrand;                        //PM
21958: 388 unit uPSI_JvSticker;                     //JCL
21959: 389 unit uPSI_XmlVerySimple;                 //mX4
21960: 390 unit uPSI_Services;                      //ExtPascal
21961: 391 unit uPSI_ExtPascalUtils;                //ExtPascal
21962: 392 unit uPSI_SocketsDelphi;                 //ExtPascal
21963: 393 unit uPSI_StBarC;                        //SysTools
21964: 394 unit uPSI_StDbBarC;                      //SysTools
21965: 395 unit uPSI_StBarPN;                       //SysTools
21966: 396 unit uPSI_StDbPNBC;                      //SysTools
21967: 397 unit uPSI_StDb2DBC;                      //SysTools
21968: 398 unit uPSI_StMoney;                       //SysTools
21969: 399 unit uPSI_JvForth;                       //JCL
21970: 400 unit uPSI_RestRequest;                   //mX4
21971: 401 unit uPSI_HttpRESTConnectionIndy;        //mX4
21972: 402 unit uPSI_JvXmlDatabase; //update       //JCL
21973: 403 unit uPSI_StAstro;                       //SysTools
21974: 404 unit uPSI_StSort;                        //SysTools
21975: 405 unit uPSI_StDate;                        //SysTools
21976: 406 unit uPSI_StDateSt;                      //SysTools
21977: 407 unit uPSI_StBase;                        //SysTools
21978: 408 unit uPSI_StVInfo;                       //SysTools
21979: 409 unit uPSI_JvBrowseFolder;                //JCL
21980: 410 unit uPSI_JvBoxProcs;                    //JCL
21981: 411 unit uPSI_urandom; (unit uranuvag;)      //DMath
21982: 412 unit uPSI_usimann; (unit ugenalg;)       //DMath
21983: 413 unit uPSI_JvHighlighter;                 //JCL
21984: 414 unit uPSI_Diff;                          //mX4
21985: 415 unit uPSI_SpringWinAPI;                  //DSpring
21986: 416 unit uPSI_StBits;                        //SysTools
21987: 417 unit uPSI_TomDBQue;                      //mX4
21988: 418 unit uPSI_MultilangTranslator;           //mX4
21989: 419 unit uPSI_HyperLabel;                    //mX4
21990: 420 unit uPSI_Starter;                       //mX4
21991: 421 unit uPSI_FileAssocs;                    //devC
21992: 422 unit uPSI_devFileMonitorX;               //devC
21993: 423 unit uPSI_devrun;                        //devC
21994: 424 unit uPSI_devExec;                       //devC
21995: 425 unit uPSI_oysUtils;                      //devC
21996: 426 unit uPSI_DosCommand;                    //devC
21997: 427 unit uPSI_CppTokenizer;                  //devC
21998: 428 unit uPSI_JvHLParser;                    //devC
21999: 429 unit uPSI_JclMapi;                       //JCL
22000: 430 unit uPSI_JclShell;                      //JCL
22001: 431 unit uPSI_JclCOM;                        //JCL
22002: 432 unit uPSI_GR32_Math;                     //Graphics32
22003: 433 unit uPSI_GR32_LowLevel;                 //Graphics32
22004: 434 unit uPSI_SimpleHl;                      //mX4
22005: 435 unit uPSI_GR32_Filters,                  //Graphics32
22006: 436 unit uPSI_GR32_VectorMaps;               //Graphics32
22007: 437 unit uPSI_cXMLFunctions;                 //Fundamentals 4
22008: 438 unit uPSI_JvTimer;                       //JCL
22009: 439 unit uPSI_cHTTPUtils;                    //Fundamentals 4
22010: 440 unit uPSI_cTLSUtils;                     //Fundamentals 4
22011: 441 unit uPSI_JclGraphics;                   //JCL
22012: 442 unit uPSI_JclSynch;                      //JCL
22013: 443 unit uPSI_IdTelnet;                      //Indy
22014: 444 unit uPSI_IdTelnetServer,                //Indy
22015: 445 unit uPSI_IdEcho,                        //Indy
22016: 446 unit uPSI_IdEchoServer,                  //Indy
22017: 447 unit uPSI_IdEchoUDP,                     //Indy
22018: 448 unit uPSI_IdEchoUDPServer,               //Indy
22019: 449 unit uPSI_IdSocks,                       //Indy
22020: 450 unit uPSI_IdAntiFreezeBase;             //Indy
22021: 451 unit uPSI_IdHostnameServer;              //Indy
22022: 452 unit uPSI_IdTunnelCommon,                //Indy
22023: 453 unit uPSI_IdTunnelMaster,                //Indy
22024: 454 unit uPSI_IdTunnelSlave,                 //Indy
22025: 455 unit uPSI_IdRSH,                         //Indy
22026: 456 unit uPSI_IdRSHServer,                   //Indy
22027: 457 unit uPSI_Spring_Cryptography_Utils;     //Spring4Delphi
22028: 458 unit uPSI_MapReader,                     //devC
22029: 459 unit uPSI_LibTar,                        //devC
22030: 460 unit uPSI_IdStack;                       //Indy
22031: 461 unit uPSI_IdBlockCipherIntercept;        //Indy
```

```
22032: 462 unit uPSI_IdChargenServer;                //Indy
22033: 463 unit uPSI_IdFTPServer,                    //Indy
22034: 464 unit uPSI_IdException,                    //Indy
22035: 465 unit uPSI_utexplot;                       //DMath
22036: 466 unit uPSI_uwinstr;                        //DMath
22037: 467 unit uPSI_VarRecUtils;                    //devC
22038: 468 unit uPSI_JvStringListToHtml,             //JCL
22039: 469 unit uPSI_JvStringHolder,                 //JCL
22040: 470 unit uPSI_IdCoder;                       //Indy
22041: 471 unit uPSI_SynHighlighterDfm;             //Synedit
22042: 472 unit uHighlighterProcs; in 471           //Synedit
22043: 473 unit uPSI_LazFileUtils,                  //LCL
22044: 474 unit uPSI_IDECmdLine;                    //LCL
22045: 475 unit uPSI_lazMasks;                      //LCL
22046: 476 unit uPSI_ip_misc;                       //mX4
22047: 477 unit uPSI_Barcode;                       //LCL
22048: 478 unit uPSI_SimpleXML;                     //LCL
22049: 479 unit uPSI_JclIniFiles;                   //JCL
22050: 480 unit uPSI_D2XXUnit;  {$X-}               //FTDI
22051: 481 unit uPSI_JclDateTime;                   //JCL
22052: 482 unit uPSI_JclEDI;                        //JCL
22053: 483 unit uPSI_JclMiscel2;                    //JCL
22054: 484 unit uPSI_JclValidation;                 //JCL
22055: 485 unit uPSI_JclAnsiStrings; {-PString}     //JCL
22056: 486 unit uPSI_SynEditMiscProcs2;             //Synedit
22057: 487 unit uPSI_JclStreams;                    //JCL
22058: 488 unit uPSI_QRCode;                        //mX4
22059: 489 unit uPSI_BlockSocket;                   //ExtPascal
22060: 490 unit uPSI_Masks,Utils                    //VCL
22061: 491 unit uPSI_synautil;                      //Synapse!
22062: 492 unit uPSI_JclMath_Class;                 //JCL RTL
22063: 493 unit ugamdist; //Gamma function         //DMath
22064: 494 unit uibeta, ucorrel; //IBeta           //DMath
22065: 495 unit uPSI_SRMgr;                         //mX4
22066: 496 unit uPSI_HotLog;                        //mX4
22067: 497 unit uPSI_DebugBox;                      //mX4
22068: 498 unit uPSI_ustrings;                      //DMath
22069: 499 unit uPSI_uregtest;                      //DMath
22070: 500 unit uPSI_usimplex;                      //DMath
22071: 501 unit uPSI_uhyper;                        //DMath
22072: 502 unit uPSI_IdHL7;                         //Indy
22073: 503 unit uPSI_IdIPMCastBase,                 //Indy
22074: 504 unit uPSI_IdIPMCastServer;               //Indy
22075: 505 unit uPSI_IdIPMCastClient;               //Indy
22076: 506 unit uPSI_unlfit; //nlregression        //DMath
22077: 507 unit uPSI_IdRawHeaders;                  //Indy
22078: 508 unit uPSI_IdRawClient;                   //Indy
22079: 509 unit uPSI_IdRawFunctions;                //Indy
22080: 510 unit uPSI_IdTCPStream;                   //Indy
22081: 511 unit uPSI_IdSNPP;                        //Indy
22082: 512 unit uPSI_St2DBarC;                      //SysTools
22083: 513 unit uPSI_ImageWin;  //FTL              //VCL
22084: 514 unit uPSI_CustomDrawTreeView; //FTL     //VCL
22085: 515 unit uPSI_GraphWin;  //FTL              //VCL
22086: 516 unit uPSI_actionMain;  //FTL            //VCL
22087: 517 unit uPSI_StSpawn;                       //SysTools
22088: 518 unit uPSI_CtlPanel;                      //VCL
22089: 519 unit uPSI_IdLPR;                         //Indy
22090: 520 unit uPSI_SockRequestInterpreter;        //Indy
22091: 521 unit uPSI_ulambert;                      //DMath
22092: 522 unit uPSI_ucholesk;                      //DMath
22093: 523 unit uPSI_SimpleDS;                      //VCL
22094: 524 unit uPSI_DBXSqlScanner;                 //VCL
22095: 525 unit uPSI_DBXMetaDataUtil;               //VCL
22096: 526 unit uPSI_Chart;                         //TEE
22097: 527 unit uPSI_TeeProcs;                      //TEE
22098: 528 unit mXBDEUtils;                         //mX4
22099: 529 unit uPSI_MDIEdit;                       //VCL
22100: 530 unit uPSI_CopyPrsr;                      //VCL
22101: 531 unit uPSI_SockApp;                       //VCL
22102: 532 unit uPSI_AppEvnts;                      //VCL
22103: 533 unit uPSI_ExtActns;                      //VCL
22104: 534 unit uPSI_TeEngine;                      //TEE
22105: 535 unit uPSI_CoolMain; //browser           //VCL
22106: 536 unit uPSI_StCRC;                         //SysTools
22107: 537 unit uPSI_StDecMth2;                     //SysTools
22108: 538 unit uPSI_frmExportMain;                 //Synedit
22109: 539 unit uPSI_SynDBEdit;                     //Synedit
22110: 540 unit uPSI_SynEditWildcardSearch;         //Synedit
22111: 541 unit uPSI_BoldComUtils;                  //BOLD
22112: 542 unit uPSI_BoldIsoDateTime;               //BOLD
22113: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils   //BOLD
22114: 544 unit uPSI_BoldXMLRequests;               //BOLD
22115: 545 unit uPSI_BoldStringList;                //BOLD
22116: 546 unit uPSI_BoldFileHandler;               //BOLD
22117: 547 unit uPSI_BoldContainers;                //BOLD
22118: 548 unit uPSI_BoldQueryUserDlg;              //BOLD
22119: 549 unit uPSI_BoldWinINet;                   //BOLD
22120: 550 unit uPSI_BoldQueue;                     //BOLD
```

```
22121: 551 unit uPSI_JvPcx;                          //JCL
22122: 552 unit uPSI_IdWhois;                        //Indy
22123: 553 unit uPSI_IdWhoIsServer;                  //Indy
22124: 554 unit uPSI_IdGopher;                       //Indy
22125: 555 unit uPSI_IdDateTimeStamp;                //Indy
22126: 556 unit uPSI_IdDayTimeServer;                //Indy
22127: 557 unit uPSI_IdDayTimeUDP;                   //Indy
22128: 558 unit uPSI_IdDayTimeUDPServer;             //Indy
22129: 559 unit uPSI_IdDICTServer;                   //Indy
22130: 560 unit uPSI_IdDiscardServer;                //Indy
22131: 561 unit uPSI_IdDiscardUDPServer;             //Indy
22132: 562 unit uPSI_IdMappedFTP;                    //Indy
22133: 563 unit uPSI_IdMappedPortTCP;                //Indy
22134: 564 unit uPSI_IdGopherServer;                 //Indy
22135: 565 unit uPSI_IdQotdServer;                   //Indy
22136: 566 unit uPSI_JvRgbToHtml;                    //JCL
22137: 567 unit uPSI_JvRemLog,                       //JCL
22138: 568 unit uPSI_JvSysComp;                      //JCL
22139: 569 unit uPSI_JvTMTL;                         //JCL
22140: 570 unit uPSI_JvWinampAPI;                    //JCL
22141: 571 unit uPSI_MSysUtils;                      //mX4
22142: 572 unit uPSI_ESBMaths;                       //ESB
22143: 573 unit uPSI_ESBMaths2;                      //ESB
22144: 574 unit uPSI_uLkJSON;                        //Lk
22145: 575 unit uPSI_ZURL;  //Zeos                   //Zeos
22146: 576 unit uPSI_ZSysUtils;                      //Zeos
22147: 577 unit unaUtils internals                   //UNA
22148: 578 unit uPSI_ZMatchPattern;                  //Zeos
22149: 579 unit uPSI_ZClasses;                       //Zeos
22150: 580 unit uPSI_ZCollections;                   //Zeos
22151: 581 unit uPSI_ZEncoding;                      //Zeos
22152: 582 unit uPSI_IdRawBase;                      //Indy
22153: 583 unit uPSI_IdNTLM;                         //Indy
22154: 584 unit uPSI_IdNNTP;                         //Indy
22155: 585 unit uPSI_usniffer; //PortScanForm        //mX4
22156: 586 unit uPSI_IdCoderMIME;                    //Indy
22157: 587 unit uPSI_IdCoderUUE;                     //Indy
22158: 588 unit uPSI_IdCoderXXE;                     //Indy
22159: 589 unit uPSI_IdCoder3to4;                    //Indy
22160: 590 unit uPSI_IdCookie;                       //Indy
22161: 591 unit uPSI_IdCookieManager;                //Indy
22162: 592 unit uPSI_WDosSocketUtils;                //WDos
22163: 593 unit uPSI_WDosPlcUtils;                   //WDos
22164: 594 unit uPSI_WDosPorts;                      //WDos
22165: 595 unit uPSI_WDosResolvers;                  //WDos
22166: 596 unit uPSI_WDosTimers;                     //WDos
22167: 597 unit uPSI_WDosPlcs;                       //WDos
22168: 598 unit uPSI_WDosPneumatics;                 //WDos
22169: 599 unit uPSI_IdFingerServer;                 //Indy
22170: 600 unit uPSI_IdDNSResolver;                  //Indy
22171: 601 unit uPSI_IdHTTPWebBrokerBridge;          //Indy
22172: 602 unit uPSI_IdIntercept;                    //Indy
22173: 603 unit uPSI_IdIPMCastBase;                  //Indy
22174: 604 unit uPSI_IdLogBase;                      //Indy
22175: 605 unit uPSI_IdIOHandlerStream;              //Indy
22176: 606 unit uPSI_IdMappedPortUDP;                //Indy
22177: 607 unit uPSI_IdQOTDUDPServer;                //Indy
22178: 608 unit uPSI_IdQOTDUDP;                      //Indy
22179: 609 unit uPSI_IdSysLog;                       //Indy
22180: 610 unit uPSI_IdSysLogServer;                 //Indy
22181: 611 unit uPSI_IdSysLogMessage;                //Indy
22182: 612 unit uPSI_IdTimeServer;                   //Indy
22183: 613 unit uPSI_IdTimeUDP;                      //Indy
22184: 614 unit uPSI_IdTimeUDPServer;                //Indy
22185: 615 unit uPSI_IdUserAccounts;                 //Indy
22186: 616 unit uPSI_TextUtils;                      //mX4
22187: 617 unit uPSI_MandelbrotEngine;               //mX4
22188: 618 unit uPSI_delphi_arduino_Unit1;           //mX4
22189: 619 unit uPSI_DTDSchema2;                     //mX4
22190: 620 unit uPSI_fplotMain;                      //DMath
22191: 621 unit uPSI_FindFileIter;                   //mX4
22192: 622 unit uPSI_PppState;  (JclStrHashMap)      //PPP
22193: 623 unit uPSI_PppParser;                      //PPP
22194: 624 unit uPSI_PppLexer;                       //PPP
22195: 625 unit uPSI_PCharUtils;                     //PPP
22196: 626 unit uPSI_uJSON;                          //WU
22197: 627 unit uPSI_JclStrHashMap;                  //JCL
22198: 628 unit uPSI_JclHookExcept;                  //JCL
22199: 629 unit uPSI_EncdDecd;                       //VCL
22200: 630 unit uPSI_SockAppReg;                     //VCL
22201: 631 unit uPSI_PJFileHandle;                   //PJ
22202: 632 unit uPSI_PJEnvVars;                      //PJ
22203: 633 unit uPSI_PJPipe;                         //PJ
22204: 634 unit uPSI_PJPipeFilters;                  //PJ
22205: 635 unit uPSI_PJConsoleApp;                   //PJ
22206: 636 unit uPSI_UConsoleAppEx;                  //PJ
22207: 637 unit uPSI_DbxSocketChannelNative,         //VCL
22208: 638 unit uPSI_DbxDataGenerator,               //VCL
22209: 639 unit uPSI_DBXClient;                      //VCL
```

```
22210: 640 unit uPSI_IdLogEvent;                    //Indy
22211: 641 unit uPSI_Reversi;                       //mX4
22212: 642 unit uPSI_Geometry;                      //mX4
22213: 643 unit uPSI_IdSMTPServer;                  //Indy
22214: 644 unit uPSI_Textures;                      //mX4
22215: 645 unit uPSI_IBX;                           //VCL
22216: 646 unit uPSI_IWDBCommon;                    //VCL
22217: 647 unit uPSI_SortGrid;                      //mX4
22218: 648 unit uPSI_IB;                            //VCL
22219: 649 unit uPSI_IBScript;                      //VCL
22220: 650 unit uPSI_JvCSVBaseControls;             //JCL
22221: 651 unit uPSI_Jvg3DColors;                   //JCL
22222: 652 unit uPSI_JvHLEditor;  //beat            //JCL
22223: 653 unit uPSI_JvShellHook;                   //JCL
22224: 654 unit uPSI_DBCommon2                      //VCL
22225: 655 unit uPSI_JvSHFileOperation;             //JCL
22226: 656 unit uPSI_uFilexport;                    //mX4
22227: 657 unit uPSI_JvDialogs;                     //JCL
22228: 658 unit uPSI_JvDBTreeView;                  //JCL
22229: 659 unit uPSI_JvDBUltimGrid;                 //JCL
22230: 660 unit uPSI_JvDBQueryParamsForm;           //JCL
22231: 661 unit uPSI_JvExControls;                  //JCL
22232: 662 unit uPSI_JvBDEMemTable;                 //JCL
22233: 663 unit uPSI_JvCommStatus;                  //JCL
22234: 664 unit uPSI_JvMailSlots2;                  //JCL
22235: 665 unit uPSI_JvgWinMask;                    //JCL
22236: 666 unit uPSI_StEclpse;                      //SysTools
22237: 667 unit uPSI_StMime;                        //SysTools
22238: 668 unit uPSI_StList;                        //SysTools
22239: 669 unit uPSI_StMerge;                       //SysTools
22240: 670 unit uPSI_StStrS;                        //SysTools
22241: 671 unit uPSI_StTree,                        //SysTools
22242: 672 unit uPSI_StVArr;                        //SysTools
22243: 673 unit uPSI_StRegIni;                      //SysTools
22244: 674 unit uPSI_urkf;                          //DMath
22245: 675 unit uPSI_usvd;                          //DMath
22246: 676 unit uPSI_DepWalkUtils;                  //JCL
22247: 677 unit uPSI_OptionsFrm;                    //JCL
22248: 678 unit yuvconverts;                        //mX4
22249: 679 uPSI_JvPropAutoSave;                     //JCL
22250: 680 uPSI_AclAPI;                             //alcinoe
22251: 681 uPSI_AviCap;                             //alcinoe
22252: 682 uPSI_ALAVLBinaryTree;                    //alcinoe
22253: 683 uPSI_ALFcnMisc;                          //alcinoe
22254: 684 uPSI_ALStringList;                       //alcinoe
22255: 685 uPSI_ALQuickSortList;                    //alcinoe
22256: 686 uPSI_ALStaticText;                       //alcinoe
22257: 687 uPSI_ALJSONDoc;                          //alcinoe
22258: 688 uPSI_ALGSMComm;                          //alcinoe
22259: 689 uPSI_ALWindows;                          //alcinoe
22260: 690 uPSI_ALMultiPartFormDataParser;          //alcinoe
22261: 691 uPSI_ALHttpCommon;                       //alcinoe
22262: 692 uPSI_ALWebSpider,                        //alcinoe
22263: 693 uPSI_ALHttpClient;                       //alcinoe
22264: 694 uPSI_ALFcnHTML;                          //alcinoe
22265: 695 uPSI_ALFTPClient;                        //alcinoe
22266: 696 uPSI_ALInternetMessageCommon;            //alcinoe
22267: 697 uPSI_ALWininetHttpClient;                //alcinoe
22268: 698 uPSI_ALWinInetFTPClient;                 //alcinoe
22269: 699 uPSI_ALWinHttpWrapper;                   //alcinoe
22270: 700 uPSI_ALWinHttpClient;                    //alcinoe
22271: 701 uPSI_ALFcnWinSock;                       //alcinoe
22272: 702 uPSI_ALFcnSQL;                           //alcinoe
22273: 703 uPSI_ALFcnCGI;                           //alcinoe
22274: 704 uPSI_ALFcnExecute;                       //alcinoe
22275: 705 uPSI_ALFcnFile;                          //alcinoe
22276: 706 uPSI_ALFcnMime;                          //alcinoe
22277: 707 uPSI_ALPhpRunner;                        //alcinoe
22278: 708 uPSI_ALGraphic;                          //alcinoe
22279: 709 uPSI_ALIniFiles;                         //alcinoe
22280: 710 uPSI_ALMemCachedClient;                  //alcinoe
22281: 711 unit uPSI_MyGrids;                       //mX4
22282: 712 uPSI_ALMultiPartMixedParser              //alcinoe
22283: 713 uPSI_ALSMTPClient                        //alcinoe
22284: 714 uPSI_ALNNTPClient;                       //alcinoe
22285: 715 uPSI_ALHintBalloon;                      //alcinoe
22286: 716 unit uPSI_ALXmlDoc;                      //alcinoe
22287: 717 unit uPSI_IPCThrd;                       //VCL
22288: 718 unit uPSI_MonForm;                       //VCL
22289: 719 unit uPSI_TeCanvas;                      //Orpheus
22290: 720 unit uPSI_Ovcmisc;                       //Orpheus
22291: 721 unit uPSI_ovcfiler;                      //Orpheus
22292: 722 unit uPSI_ovcstate;                      //Orpheus
22293: 723 unit uPSI_ovccoco;                       //Orpheus
22294: 724 unit uPSI_ovcrvexp;                      //Orpheus
22295: 725 unit uPSI_OvcFormatSettings;             //Orpheus
22296: 726 unit uPSI_OvcUtils;                      //Orpheus
22297: 727 unit uPSI_ovcstore;                      //Orpheus
22298: 728 unit uPSI_ovcstr;                        //Orpheus
```

```
22299: 729 unit uPSI_ovcmru;                          //Orpheus
22300: 730 unit uPSI_ovccmd;                          //Orpheus
22301: 731 unit uPSI_ovctimer;                        //Orpheus
22302: 732 unit uPSI_ovcintl;                         //Orpheus
22303: 733 uPSI_AfCircularBuffer;                     //AsyncFree
22304: 734 uPSI_AfUtils;                              //AsyncFree
22305: 735 uPSI_AfSafeSync;                           //AsyncFree
22306: 736 uPSI_AfComPortCore;                        //AsyncFree
22307: 737 uPSI_AfComPort;                            //AsyncFree
22308: 738 uPSI_AfPortControls;                       //AsyncFree
22309: 739 uPSI_AfDataDispatcher;                     //AsyncFree
22310: 740 uPSI_AfViewers;                            //AsyncFree
22311: 741 uPSI_AfDataTerminal;                       //AsyncFree
22312: 742 uPSI_SimplePortMain;                       //AsyncFree
22313: 743 unit uPSI_ovcclock;                        //Orpheus
22314: 744 unit uPSI_o32intlst;                       //Orpheus
22315: 745 unit uPSI_o32ledlabel;                     //Orpheus
22316: 746 unit uPSI_AlMySqlClient;                   //alcinoe
22317: 747 unit uPSI_ALFBXClient;                     //alcinoe
22318: 748 unit uPSI_ALFcnSQL;                        //alcinoe
22319: 749 unit uPSI_AsyncTimer;                      //mX4
22320: 750 unit uPSI_ApplicationFileIO;               //mX4
22321: 751 unit uPSI_PsAPI;                           //VCLé
22322: 752 uPSI_ovcuser;                              //Orpheus
22323: 753 uPSI_ovcurl;                               //Orpheus
22324: 754 uPSI_ovcvlb;                               //Orpheus
22325: 755 uPSI_ovccolor;                             //Orpheus
22326: 756 uPSI_ALFBXLib,                             //alcinoe
22327: 757 uPSI_ovcmeter;                             //Orpheus
22328: 758 uPSI_ovcpeakm;                             //Orpheus
22329: 759 uPSI_O32BGSty;                             //Orpheus
22330: 760 uPSI_ovcBidi;                              //Orpheus
22331: 761 uPSI_ovctcary;                             //Orpheus
22332: 762 uPSI_DXPUtils;                             //mX4
22333: 763 uPSI_ALMultiPartBaseParser;                //alcinoe
22334: 764 uPSI_ALMultiPartAlternativeParser;         //alcinoe
22335: 765 uPSI_ALPOP3Client;                         //alcinoe
22336: 766 uPSI_SmallUtils;                           //mX4
22337: 767 uPSI_MakeApp;                              //mX4
22338: 768 uPSI_O32MouseMon;                          //Orpheus
22339: 769 uPSI_OvcCache;                             //Orpheus
22340: 770 uPSI_ovccalc;                              //Orpheus
22341: 771 uPSI_Joystick,                             //OpenGL
22342: 772 uPSI_ScreenSaver;                          //OpenGL
22343: 773 uPSI_XCollection,                          //OpenGL
22344: 774 uPSI_Polynomials,                          //OpenGL
22345: 775 uPSI_PersistentClasses, //9.86             //OpenGL
22346: 776 uPSI_VectorLists;                          //OpenGL
22347: 777 uPSI_XOpenGL,                             //OpenGL
22348: 778 uPSI_MeshUtils;                           //OpenGL
22349: 779 unit uPSI_JclSysUtils;                     //JCL
22350: 780 unit uPSI_JclBorlandTools;                 //JCL
22351: 781 unit JclFileUtils_max;                     //JCL
22352: 782 uPSI_AfDataControls,                       //AsyncFree
22353: 783 uPSI_GLSilhouette;                         //OpenGL
22354: 784 uPSI_JclSysUtils_class;                    //JCL
22355: 785 uPSI_JclFileUtils_class;                   //JCL
22356: 786 uPSI_FileUtil;                             //JCL
22357: 787 uPSI_changefind;                           //mX4
22358: 788 uPSI_cmdIntf;                              //mX4
22359: 789 uPSI_fservice;                             //mX4
22360: 790 uPSI_Keyboard;                             //OpenGL
22361: 791 uPSI_VRMLParser,                           //OpenGL
22362: 792 uPSI_GLFileVRML,                           //OpenGL
22363: 793 uPSI_Octree;                               //OpenGL
22364: 794 uPSI_GLPolyhedron,                         //OpenGL
22365: 795 uPSI_GLCrossPlatform;                      //OpenGL
22366: 796 uPSI_GLParticles;                          //OpenGL
22367: 797 uPSI_GLNavigator;                          //OpenGL
22368: 798 uPSI_GLStarRecord;                         //OpenGL
22369: 799 uPSI_GLTextureCombiners;                   //OpenGL
22370: 800 uPSI_GLCanvas;                             //OpenGL
22371: 801 uPSI_GeometryBB;                           //OpenGL
22372: 802 uPSI_GeometryCoordinates;                  //OpenGL
22373: 803 uPSI_VectorGeometry;                       //OpenGL
22374: 804 uPSI_BumpMapping;                          //OpenGL
22375: 805 uPSI_TGA;                                  //OpenGL
22376: 806 uPSI_GLVectorFileObjects;                  //OpenGL
22377: 807 uPSI_IMM;                                  //VCL
22378: 808 uPSI_CategoryButtons;                      //VCL
22379: 809 uPSI_ButtonGroup;                          //VCL
22380: 810 uPSI_DbExcept;                             //VCL
22381: 811 uPSI_AxCtrls;                              //VCL
22382: 812 uPSI_GL_actorUnit1;                        //OpenGL
22383: 813 uPSI_StdVCL;                               //VCL
22384: 814 unit CurvesAndSurfaces;                    //OpenGL
22385: 815 uPSI_DataAwareMain;                        //AsyncFree
22386: 816 uPSI_TabNotBk;                             //VCL
22387: 817 uPSI_udwsfiler;                            //mX4
```

```
22388: 818 uPSI_synaip;                          //Synapse!
22389: 819 uPSI_synacode;                        //Synapse
22390: 820 uPSI_synachar;                        //Synapse
22391: 821 uPSI_synamisc;                        //Synapse
22392: 822 uPSI_synaser;                        //Synapse
22393: 823 uPSI_synaicnv;                        //Synapse
22394: 824 uPSI_tlntsend;                       //Synapse
22395: 825 uPSI_pingsend;                       //Synapse
22396: 826 uPSI_blcksock;                       //Synapse
22397: 827 uPSI_asn1util;                       //Synapse
22398: 828 uPSI_dnssend;                        //Synapse
22399: 829 uPSI_clamsend;                       //Synapse
22400: 830 uPSI_ldapsend;                       //Synapse
22401: 831 uPSI_mimemess;                       //Synapse
22402: 832 uPSI_slogsend;                       //Synapse
22403: 833 uPSI_mimepart;                       //Synapse
22404: 834 uPSI_mimeinln;                       //Synapse
22405: 835 uPSI_ftpsend,                        //Synapse
22406: 836 uPSI_ftptsend;                       //Synapse
22407: 837 uPSI_httpsend;                       //Synapse
22408: 838 uPSI_sntpsend;                       //Synapse
22409: 839 uPSI_smtpsend;                       //Synapse
22410: 840 uPSI_snmpsend;                       //Synapse
22411: 841 uPSI_imapsend;                       //Synapse
22412: 842 uPSI_pop3send;                       //Synapse
22413: 843 uPSI_nntpsend;                       //Synapse
22414: 844 uPSI_ssl_cryptlib;                   //Synapse
22415: 845 uPSI_ssl_openssl;                    //Synapse
22416: 846 uPSI_synhttp_daemon;                 //Synapse
22417: 847 uPSI_NetWork;                        //mX4
22418: 848 uPSI_PingThread;                     //Synapse
22419: 849 uPSI_JvThreadTimer;                  //JCL
22420: 850 unit uPSI_wwSystem;                  //InfoPower
22421: 851 unit uPSI_IdComponent;               //Indy
22422: 852 unit uPSI_IdIOHandlerThrottle;       //Indy
22423: 853 unit uPSI_Themes;                    //VCL
22424: 854 unit uPSI_StdStyleActnCtrls;         //VCL
22425: 855 unit uPSI_UDDIHelper;                //VCL
22426: 856 unit uPSI_IdIMAP4Server;             //Indy
22427: 857 uPSI_VariantSymbolTable,             //VCL //3.9.9.92
22428: 858 uPSI_udf_glob,                       //mX4
22429: 859 uPSI_TabGrid,                        //VCL
22430: 860 uPSI_JsDBTreeView,                   //mX4
22431: 861 uPSI_JsSendMail,                     //mX4
22432: 862 uPSI_dbTvRecordList,                 //mX4
22433: 863 uPSI_TreeVwEx,                       //mX4
22434: 864 uPSI_ECDataLink,                     //mX4
22435: 865 uPSI_dbTree,                         //mX4
22436: 866 uPSI_dbTreeCBox,                     //mX4
22437: 867 unit uPSI_Debug; //TfrmDebug        //mX4
22438: 868 uPSI_TimeFncs;                       //mX4
22439: 869 uPSI_FileIntf,                       //VCL
22440: 870 uPSI_SockTransport,                  //RTL
22441: 871 unit uPSI_WinInet;                   //RTL
22442: 872 unit uPSI_Wwstr;                     //mX4
22443: 873 uPSI_DBLookup,                       //VCL
22444: 874 uPSI_Hotspot,                        //mX4
22445: 875 uPSI_HList; //History List           //mX4
22446: 876 unit uPSI_DrTable;                   //VCL
22447: 877 uPSI_TConnect,                       //VCL
22448: 878 uPSI_DataBkr,                        //VCL
22449: 879 uPSI_HTTPIntr;                       //VCL
22450: 880 unit uPSI_Mathbox;                   //mX4
22451: 881 uPSI_cyIndy,                         //cY
22452: 882 uPSI_cySysUtils,                     //cY
22453: 883 uPSI_cyWinUtils,                     //cY
22454: 884 uPSI_cyStrUtils,                     //cY
22455: 885 uPSI_cyObjUtils,                     //cY
22456: 886 uPSI_cyDateUtils,                    //cY
22457: 887 uPSI_cyBDE,                          //cY
22458: 888 uPSI_cyClasses,                      //cY
22459: 889 uPSI_cyGraphics,  //3.9.9.94_2      //cY
22460: 890 unit uPSI_cyTypes;                   //cY
22461: 891 uPSI_JvDateTimePicker,               //JCL
22462: 892 uPSI_JvCreateProcess,                //JCL
22463: 893 uPSI_JvEasterEgg,                    //JCL
22464: 894 uPSI_WinSvc,  //3.9.9.94_3          //VCL
22465: 895 uPSI_SvcMgr                          //VCL
22466: 896 unit uPSI_JvPickDate;                //JCL
22467: 897 unit uPSI_JvNotify;                  //JCL
22468: 898 uPSI_JvStrHlder                      //JCL
22469: 899 unit uPSI_JclNTFS2;                  //JCL
22470: 900 uPSI_Jcl8087 //math coprocessor      //JCL
22471: 901 uPSI_JvAddPrinter                    //JCL
22472: 902 uPSI_JvCabFile                       //JCL
22473: 903 uPSI_JvDataEmbedded;                 //JCL
22474: 904 unit uPSI_U_HexView;                 //mX4
22475: 905 uPSI_UWavein4,                       //mX4
22476: 906 uPSI_AMixer,                         //mX4
```

```
22477: 907 uPSI_JvaScrollText,                    //mX4
22478: 908 uPSI_JvArrow,                          //mX4
22479: 909 unit uPSI_UrlMon;                      //mX4
22480: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'  //mX4
22481: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;  //DFF
22482: 912 unit uPSI_DFFUtils;                     //DFF
22483: 913 unit uPSI_MathsLib;                     //DFF
22484: 914 uPSI_UIntList;                         //DFF
22485: 915 uPSI_UGetParens;                       //DFF
22486: 916 unit uPSI_UGeometry;                   //DFF
22487: 917 unit uPSI_UAstronomy;                  //DFF
22488: 918 unit uPSI_UCardComponentV2;            //DFF
22489: 919 unit uPSI_UTGraphSearch;               //DFF
22490: 920 unit uPSI_UParser10;                   //DFF
22491: 921 unit uPSI_cyIEUtils;                   //cY
22492: 922 unit uPSI_UcomboV2;                    //DFF
22493: 923 uPSI_cyBaseComm,                       //cY
22494: 924 uPSI_cyAppInstances,                   //cY
22495: 925 uPSI_cyAttract,                        //cY
22496: 926 uPSI_cyDERUtils                        //cY
22497: 927 unit uPSI_cyDocER;                     //cY
22498: 928 unit uPSI_ODBC;                        //mX
22499: 929 unit uPSI_AssocExec;                   //mX
22500: 930 uPSI_cyBaseCommRoomConnector,          //cY
22501: 931 uPSI_cyCommRoomConnector,              //cY
22502: 932 uPSI_cyCommunicate,                    //cY
22503: 933 uPSI_cyImage;                          //cY
22504: 934 uPSI_cyBaseContainer                   //cY
22505: 935 uPSI_cyModalContainer,                 //cY
22506: 936 uPSI_cyFlyingContainer;                //cY
22507: 937 uPSI_RegStr,                           //VCL
22508: 938 uPSI_HtmlHelpViewer;                   //VCL
22509: 939 unit uPSI_cyIniForm                    //cY
22510: 940 unit uPSI_cyVirtualGrid;               //cY
22511: 941 uPSI_Profiler,                         //DA
22512: 942 uPSI_BackgroundWorker,                 //DA
22513: 943 uPSI_WavePlay,                         //DA
22514: 944 uPSI_WaveTimer,                        //DA
22515: 945 uPSI_WaveUtils;                        //DA
22516: 946 uPSI_NamedPipes,                       //TB
22517: 947 uPSI_NamedPipeServer,                  //TB
22518: 948 unit uPSI_process,                     //TB
22519: 949 unit uPSI_DPUtils;                     //TB
22520: 950 unit uPSI_CommonTools;                 //TB
22521: 951 uPSI_DataSendToWeb,                    //TB
22522: 952 uPSI_StarCalc,                         //TB
22523: 953 uPSI_D2_XPVistaHelperU                 //TB
22524: 954 unit uPSI_NetTools                     //TB
22525: 955 unit uPSI_Pipes                        //TB
22526: 956 uPSI_ProcessUnit,                      //mX
22527: 957 uPSI_adGSM,                            //mX
22528: 958 unit uPSI_BetterADODataSet;            //mX
22529: 959 unit uPSI_AdSelCom; //FTT             //mX
22530: 960 unit unit uPSI_dwsXPlatform;          //DWS
22531: 961 uPSI_AdSocket;                        //mX Turbo Power
22532: 962 uPSI_AdPacket;                        //mX
22533: 963 uPSI_AdPort;                          //mX
22534: 964 uPSI_PathFunc;                        //Inno
22535: 965 uPSI_CmnFunc;                         //Inno
22536: 966 uPSI_CmnFunc2; //Inno Setup           //Inno
22537: 967 unit uPSI_BitmapImage;                //mX4
22538: 968 unit uPSI_ImageGrabber;               //mX4
22539: 969 uPSI_SecurityFunc,                    //Inno
22540: 970 uPSI_RedirFunc,                       //Inno
22541: 971 uPSI_FIFO, (MemoryStream)             //mX4
22542: 972 uPSI_Int64Em,                         //Inno
22543: 973 unit uPSI_InstFunc;                   //Inno
22544: 974 unit uPSI_LibFusion;                  //Inno
22545: 975 uPSI_SimpleExpression;                //Inno
22546: 976 uPSI_unitResourceDetails,             //XN
22547: 977 uPSI_unitResFile,                     //XN
22548: 978 unit uPSI_simpleComport;              //mX4
22549: 979 unit uPSI_AfViewershelpers;           //Async
22550: 980 unit uPSI_Console;                    //mX4
22551: 981 unit uPSI_AnalogMeter;                //TB
22552: 982 unit uPSI_XPrinter,                   //TB
22553: 983 unit uPSI_IniFiles;                   //VCL
22554: 984 unit uPSI_lazIniFiles;                //FP
22555: 985 uPSI_testutils;                       //FP
22556: 986 uPSI_ToolsUnit; (DBTests)             //FP
22557: 987 uPSI_fpcunit                          //FP
22558: 988 uPSI_testdecorator;                   //FP
22559: 989 unit uPSI_fpcunittests;               //FP
22560: 990 unit uPSI_cTCPBuffer;                 //Fundamentals 4
22561: 991 unit uPSI_Glut,                       //Open GL
22562: 992 uPSI_LEDBitmaps,                      //mX4
22563: 993 uPSI_FileClass,                       //Inno
22564: 994 uPSI_FileUtilsClass,                  //mX4
22565: 995 uPSI_ComPortInterface; //Kit          //mX4
```

```
22566: 996 unit uPSI_SwitchLed;                      //mX4
22567: 997 unit uPSI_cyDmmCanvas;                    //cY
22568: 998 uPSI_uColorFunctions;                     //DFF
22569: 999 uPSI_uSettings;                           //DFF
22570: 1000 uPSI_cyDebug.pas                         //cY
22571: 1001 uPSI_cyColorMatrix;                      //cY
22572: 1002 unit uPSI_cyCopyFiles;                   //cY
22573: 1003 unit uPSI_cySearchFiles;                 //cY
22574: 1004 unit uPSI_cyBaseMeasure;                 //cY
22575: 1005 unit uPSI_PJIStreams;                    //DD
22576: 1006 unit uPSI_cyRunTimeResize;               //cY
22577: 1007 unit uPSI_jcontrolutils;                 //cY
22578: 1008 unit uPSI_kcMapViewer; (+GEONames)      //kc
22579: 1009 unit uPSI_kcMapViewerDESynapse;         //kc
22580: 1010 unit uPSI_cparserutils; (+GIS_SysUtils) //kc
22581: 1011 unit uPSI_LedNumber;                    //TurboPower
22582: 1012 unit uPSI_StStrL;                       //SysTools
22583: 1013 unit uPSI_indGnouMeter;                 //LAZ
22584: 1014 unit uPSI_Sensors;                      //LAZ
22585: 1015 unit uPSI_pwmain;  //cgi of powtils    //Pow
22586: 1016 unit uPSI_HTMLUtil;                     //Pow
22587: 1017 unit uPSI_synwrap1; //httpsend         //Pow
22588: 1018 unit StreamWrap1                        //Pow
22589: 1019 unit uPSI_pwmain;                       //Pow
22590: 1020 unit pwtypes                            //Pow
22591: 1021 uPSI_W32VersionInfo                     //LAZ
22592: 1022 unit uPSI_IpAnim;                       //LAZ
22593: 1023 unit uPSI_IpUtils; //iputils2(TurboPower) //TP
22594: 1024 unit uPSI_LrtPoTools;                   //LAZ
22595: 1025 unit uPSI_Laz_DOM;                      //LAZ
22596: 1026 unit uPSI_hhAvComp;                     //LAZ
22597: 1027 unit uPSI_GPS2;
22598: 1028 unit uPSI_GPS;
22599: 1029 unit uPSI_GPSUDemo; // formtemplate TFDemo
22600: 1030 unit uPSI_NMEA;     // GPS
22601: 1031 unit uPSI_ScreenThreeDLab;
22602:
22603:
22604:
22605: ////////////////////////////////////////////////////////////////////////////
22606: //Form Template Library FTL
22607: ////////////////////////////////////////////////////////////////////////////
22608:
22609: 34 FTL For Form Building out of the Script, eg. 399_form_templates.txt
22610:
22611: 045 unit uPSI_VListView                      TFormListView;
22612: 263 unit uPSI_JvProfiler32;                  TProfReport
22613: 270 unit uPSI_ImgList;                       TCustomImageList
22614: 278 unit uPSI_JvImageWindow;                 TJvImageWindow
22615: 317 unit uPSI_JvParserForm;                  TJvHTMLParserForm
22616: 497 unit uPSI_DebugBox;                      TDebugBox
22617: 513 unit uPSI_ImageWin;                      TImageForm, TImageForm2
22618: 514 unit uPSI_CustomDrawTreeView;           TCustomDrawForm
22619: 515 unit uPSI_GraphWin;                      TGraphWinForm
22620: 516 unit uPSI_actionMain;                   TActionForm
22621: 518 unit uPSI_CtlPanel;                     TAppletApplication
22622: 529 unit uPSI_MDIEdit;                      TEditForm
22623: 535 unit uPSI_CoolMain; {browser}          TWebMainForm
22624: 538 unit uPSI_frmExportMain;               TSynexportForm
22625: 585 unit uPSI_usniffer; {//PortScanForm}   TSniffForm
22626: 600 unit uPSI_ThreadForm;                  TThreadSortForm;
22627: 618 unit uPSI_delphi_arduino_Unit1;        TLEDForm
22628: 620 unit uPSI_fplotMain;                   TfplotForm1
22629: 660 unit uPSI_JvDBQueryParamsForm;         TJvQueryParamsDialog
22630: 677 unit uPSI_OptionsFrm;                  TfrmOptions;
22631: 718 unit uPSI_MonForm;                     TMonitorForm
22632: 742 unit uPSI_SimplePortMain;              TPortForm1
22633: 770 unit uPSI_ovccalc;                     TOvcCalculator  //widget
22634: 810 unit uPSI_DbExcept;                    TDbEngineErrorDlg
22635: 812 unit uPSI_GL_actorUnit1;               TglActorForm1  //OpenGL Robot
22636: 846 unit uPSI_synhttp_daemon;              TTCPHttpDaemon, TTCPHttpThrd, TPingThread
22637: 867 unit uPSI_Debug;                       TfrmDebug
22638: 904 unit uPSI_U_HexView;                   THexForm2
22639: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)  TOscfrmMain
22640: 959 unit uPSI_AdSelCom;                    TComSelectForm
22641: 1029 unit uPSI_GPSUDemo;                   TFDemo
22642: 1031 unit uPSI_ScreenThreeDLab;            TFormLab3D
22643:
22644:
22645: ex.:with TEditForm.create(self) do begin
22646:       caption:= 'Template Form Tester';
22647:       FormStyle:= fsStayOnTop;
22648:       with editor do begin
22649:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
22650:         SelStart:= 0;
22651:         Modified:= False;
22652:       end;
22653:     end;
22654:   with TWebMainForm.create(self) do begin
```

```
22655:     URLs.Text:= 'http://www.kleiner.ch';
22656:     URLsClick(self); Show;
22657:   end;
22658:   with TSynexportForm.create(self) do begin
22659:     Caption:= 'Synexport HTML RTF tester';
22660:     Show;
22661:   end;
22662:   with TThreadSortForm.create(self) do begin
22663:      showmodal; free;
22664:   end;
22665:   with TCustomDrawForm.create(self) do begin
22666:        width:=820; height:=820;
22667:        image1.height:= 600; //add properties
22668:        image1.picture.bitmap:= image2.picture.bitmap;
22669:        //SelectionBackground1Click(self) CustomDraw1Click(self);
22670:        Background1.click;
22671:        bitmap1.click; Tile1.click;
22672:        Showmodal;
22673:        Free;
22674:      end;
22675:    with TfplotForm1.Create(self) do begin
22676:      BtnPlotClick(self);
22677:      Showmodal; Free;
22678:    end;
22679:   with TOvcCalculator.create(self) do begin
22680:     parent:= aForm;
22681:    //free;
22682:     setbounds(550,510,200,150);
22683:     displaystr:= 'maXcalc';
22684:   end;
22685:   with THexForm2.Create(self) do begin
22686:      ShowModal;
22687:      Free;
22688:   end;
22689:
22690: function CheckBox: string;
22691: var idHTTP: TIDHTTP;
22692:  begin
22693:    result:= 'version not found';
22694:    if IsInternet then begin
22695:      idHTTP:= TIdHTTP.Create(NIL);
22696:      try
22697:        result:= idHTTP.Get(MXVERSIONFILE2);
22698:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
22699:        if result = MBVER2 then begin
22700:          //output.Font.Style:= [fsbold];
22701:          //Speak(' A new Version '+vstr+' of max box is available! ');
22702:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
22703:        end;
22704:       //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
22705:      finally
22706:        idHTTP.Free
22707:    end;
22708:  end;
22709: end;
22710:
22711: function ApWinExecAndWait32(FileName:PChar; CommandLine:PChar; Visibility:Integer):Integer;
22712: function KillTask(ExeFileName: string): Integer;
22713: procedure KillProcess(hWindowHandle: HWND);
22714: function FindWindowByTitle(WindowTitle: string): Hwnd;
22715:
22716:
22717: ////////////////////////////////////////////////////////////////////////////
22718: All maXbox Tutorials Table of Content 2014
22719: ////////////////////////////////////////////////////////////////////////////
22720:     Tutorial 00 Function-Coding  (Blix the Programmer)
22721:     Tutorial 01 Procedural-Coding
22722:     Tutorial 02 OO-Programming
22723:     Tutorial 03 Modular Coding
22724:     Tutorial 04 UML Use Case Coding
22725:     Tutorial 05 Internet Coding
22726:     Tutorial 06 Network Coding
22727:     Tutorial 07 Game Graphics Coding
22728:     Tutorial 08 Operating System Coding
22729:     Tutorial 09 Database Coding
22730:     Tutorial 10 Statistic Coding
22731:     Tutorial 11 Forms Coding
22732:     Tutorial 12 SQL DB Coding
22733:     Tutorial 13 Crypto Coding
22734:     Tutorial 14 Parallel Coding
22735:     Tutorial 15 Serial RS232 Coding
22736:     Tutorial 16 Event Driven Coding
22737:     Tutorial 17 Web Server Coding
22738:     Tutorial 18 Arduino System Coding
22739:     Tutorial 18_3 RGB LED System Coding
22740:     Tutorial 19 WinCOM /Arduino Coding
22741:     Tutorial 20 Regular Expressions RegEx
22742:     Tutorial 21 Android Coding (coming 2013)
22743:     Tutorial 22 Services Programming
```

```
22744:     Tutorial 23 Real Time Systems
22745:     Tutorial 24 Clean Code
22746:     Tutorial 25 maXbox Configuration I+II
22747:     Tutorial 26 Socket Programming with TCP
22748:     Tutorial 27 XML & TreeView
22749:     Tutorial 28 DLL Coding (available)
22750:     Tutorial 29 UML Scripting (2014)
22751:     Tutorial 30 Web of Things (2014)
22752:     Tutorial 31 Closures (2014)
22753:     Tutorial 32 SQL Firebird (coming 2014)
22754:     Tutorial 33 Oscilloscope (coming 2015)
22755:     Tutorial 34 GPS Navigation (2014)
22756:     Tutorial 35 Web Box (available)
22757:     Tutorial 36 Unit Testing (coming 2015)
22758:     Tutorial 37 API Coding (coming 2015)
22759:     Tutorial 38 3D Coding (coming 2015)
22760:     Tutorial 39 GEO Map Coding (available)
22761:     Tutorial 39_1 GEO Map Layers Coding (available)
22762:     Tutorial 40 REST Coding (coming 2015)
22763:
22764:
22765: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
22766: using Docu for this file is maxbox_functions_all.pdf
22767: PEP - Pascal Education Program  Low Lib Lab ShellHell
22768:
22769: http://stackoverflow.com/tags/pascalscript/hot
22770: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
22771: http://sourceforge.net/projects/maXbox   #locs:51620
22772: http://sourceforge.net/apps/mediawiki/maXbox
22773: http://www.blaisepascal.eu/
22774: https://github.com/maxkleiner/maXbox3.git
22775: http://www.heise.de/download/maxbox-1176464.html
22776: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
22777: https://www.facebook.com/pages/Programming-maXbox/166844836691703
22778: http://www.softwareschule.ch/arduino_training.pdf
22779: http://www.delphiarea.com
22780: http://www.freepascal.org/docs-html/rtl/strutils/index-5.html
22781:
22782:
22783: All maXbox Examples List
22784: https://github.com/maxkleiner/maXbox3/releases
22785: ********************************************************************************************
22786: 000_pas_baseconvert.txt                          282_fadengraphik.txt
22787: 000_pas_baseconvert.txt_encrypt                  283_SQL_API_messagetimeout.txt
22788: 000_pas_baseconvert.txt_decrypt                  284_SysTools4.txt
22789: 001_1_pas_functest - Kopie.txt                   285_MineForm_GR32.TXT
22790: 001_1_pas_functest.txt                          285_MineForm_GR32main.TXT
22791: 001_1_pas_functest2.txt                         285_MineForm_GR32mainsolution.TXT
22792: 001_1_pas_functest_clx2.txt                     285_MineForm_propas.TXT
22793: 001_1_pas_functest_clx2_2.txt                   285_MineForm_propas2.TXT
22794: 001_1_pas_functest_openarray.txt                285_minesweeper2.TXT
22795: 001_pas_lottogen.txt                            285_Patterns_process.txt
22796: 001_pas_lottogen_template.txt                   286_colormixer_jpeg_charcounter.txt
22797: 001_pas_lottogen.txtcopy                        286_colormixer_jpeg_charcounter2.txt
22798: 002_pas_russianroulette.txt                     287_eventhandling.txt
22799: 002_pas_russianroulette.txtcopy                 287_eventhandling2.txt
22800: 002_pas_russianroulette.txtcopy_decrypt         287_eventhandling2_negpower.txt
22801: 002_pas_russianroulette.txtcopy_encrypt         288_bitblt.txt
22802: 003_pas_motion.txt                              288_bitblt_resize.txt
22803: 003_pas_motion.txtcopy                          289_regression.txt
22804: 004_pas_search.txt                              289_regression2.txt
22805: 004_pas_search_replace.txt                      290_bestofbox.txt
22806: 004_search_replace_allfunctionlist.txt          290_bestofbox2.txt
22807: 005_pas_oodesign.txt                            290_bestofbox3.txt
22808: 005_pas_shelllink.txt                           291_3sort_visual_thread.txt
22809: 006_pas_oobatch.txt                             292_refactoring2.txt
22810: 007_pas_streamcopy.txt                          293_bold_utils.txt
22811: 008_EINMALEINS_FUNC.TXT                         293_ib_utils.txt
22812: 008_explanation.txt                             293_ib_utils_timetest.txt
22813: 008_pas_verwechselt.txt                         294_maxcalc_demo.txt
22814: 008_pas_verwechselt_ibz_bern_func.txt           294_maxcalc_demo2.txt
22815: 008_stack_ibz.TXT                               295_easter_calendar.txt
22816: 009_pas_umlrunner.txt                           295_easter_calendar2.txt
22817: 009_pas_umlrunner_all.txt                       295_easter_combobox.txt
22818: 009_pas_umlrunner_componenttest.txt             297_atomimage.txt
22819: 009_pas_umlrunner_solution.txt                  297_atomimage2.txt
22820: 009_pas_umlrunner_solution_2step.txt            297_atomimage3.txt
22821: 010_pas_oodesign_solution.txt                   297_atomimage4.txt
22822: 011_pas_puzzlepas_defect.txt                    297_maxonmotor.TXT
22823: 012_pas_umlrunner_solution.txt                  297_maxon_atomimage9.txt
22824: 012_pas_umlrunner_solution2.txt                 298_bitblt_animation.txt
22825: 013_pas_linenumber.txt                          298_bitblt_animation2.txt
22826: 014_pas_primetest.txt                           298_bitblt_animation3.txt
22827: 014_pas_primetest_first.txt                     298_bitblt_animation4.txt
22828: 014_pas_primetest_sync.txt                      298_bitblt_animation4_screensaver.txt
22829: 015_pas_designbycontract.txt                    298_bitblt_animation5_screensaver.txt
22830: 015_pas_designbycontract_solution.txt           299_animation.txt
22831: 016_pas_searchrec.txt                           299_animationmotor_arduino.txt
22832: 017_chartgen.txt                                299_animation_formprototype.txt
```

22833: 018_data_simulator.txt
22834: 019_dez_to_bin.txt
22835: 019_dez_to_bin_grenzwert_ibz.txt
22836: 020_proc_feedback.txt
22837: 021_pas_symkey.txt
22838: 021_pas_symkey_solution.txt
22839: 022_pas_filestreams.txt
22840: 023_pas_find_searchrec.txt
22841: 023_pas_pathfind.txt
22842: 024_pas_TFileStream_records.txt
22843: 025_prime_direct.txt
22844: 026_pas_memorystream.txt
22845: 027_pas_shellexecute_beta.txt
22846: 027_pas_shellexecute_solution.txt
22847: 028_pas_dataset.txt
22848: 029_pas_assignfile.txt
22849: 029_pas_assignfile_dragndropexe.txt
22850: 030_palindrome_2.txt
22851: 030_palindrome_tester.txt
22852: 030_pas_recursion.txt
22853: 030_pas_recursion2.txt
22854: 031_pas_hashcode.txt
22855: 032_pas_crc_const.txt
22856: 033_pas_cipher.txt
22857: 033_pas_cipher_def.txt
22858: 033_pas_cipher_file_2_solution.txt
22859: 034_pas_soundbox.txt
22860: 035_pas_crcscript.txt
22861: 035_pas_CRCscript_modbus.txt
22862: 036_pas_includetest.txt
22863: 036_pas_includetest_basta.txt
22864: 037_pas_define_demo32.txt
22865: 038_pas_box_demonstrator.txt
22866: 039_pas_dllcall.txt
22867: 040_paspointer.txt
22868: 040_paspointer_old.txt
22869: 041_pasplotter.txt
22870: 041_pasplotter_plus.txt
22871: 042_pas_kgv_ggt.txt
22872: 043_pas_proceduretype.txt
22873: 044_pas_14queens_solwith14.txt
22874: 044_pas_8queens.txt
22875: 044_pas_8queens_sol2.txt
22876: 044_pas_8queens_solutions.txt
22877: 044_queens_performer.txt
22878: 044_queens_performer2.txt
22879: 044_queens_performer2tester.txt
22880: 045_pas_listhandling.txt
22881: 046_pas_records.txt
22882: 047_pas_modula10.txt
22883: 048_pas_romans.txt
22884: 049_pas_ifdemo.txt
22885: 049_pas_ifdemo_BROKER.txt
22886: 050_pas_primetest2.txt
22887: 050_pas_primetester_thieves.txt
22888: 050_program_starter.txt
22889: 050_program_starter_performance.txt
22890: 051_pas_findtext_solution.txt
22891: 052_pas_text_as_stream.txt
22892: 052_pas_text_as_stream_include.txt
22893: 053_pas_singleton.txt
22894: 054_pas_speakpassword.txt
22895: 054_pas_speakpassword2.txt
22896: 054_pas_speakpassword_searchtest.txt
22897: 055_pas_factorylist.txt
22898: 056_pas_demeter.txt
22899: 057_pas_dirfinder.txt
22900: 058_pas_filefinder.txt
22901: 058_pas_filefinder_pdf.txt
22902: 058_pas_filefinder_screview.txt
22903: 058_pas_filefinder_screview2.txt
22904: 058_pas_filefinder_screview3.txt
22905: 059_pas_timertest.txt
22906: 059_pas_timertest_2.txt
22907: 059_pas_timertest_time_solution.txt
22908: 059_timerobject_starter2.txt
22909: 059_timerobject_starter2_ibz2_async.txt
22910: 059_timerobject_starter2_uml.txt
22911: 059_timerobject_starter2_uml_main.txt
22912: 059_timerobject_starter4_ibz.txt
22913: 060_pas_datefind.txt
22914: 060_pas_datefind_exceptions2.txt
22915: 060_pas_datefind_exceptions_CHECKTEST.txt
22916: 060_pas_datefind_fulltext.txt
22917: 060_pas_datefind_plus.txt
22918: 060_pas_datefind_plus_mydate.txt
22919: 061_pas_randomwalk.txt
22920: 061_pas_randomwalk_plus.txt
22921: 062_paskorrelation.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
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

22922: 063_pas_calculateform.txt
22923: 063_pas_calculateform_2list.txt
22924: 064_pas_timetest.txt
22925: 065_pas_bitcounter.txt
22926: 066_pas_eliza.txt
22927: 066_pas_eliza_include_sol.txt
22928: 067_pas_morse.txt
22929: 068_pas_piezo_sound.txt
22930: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
22931: 069_my_LEDBOX.TXT
22932: 069_pas_ledmatrix.txt
22933: 069_pas_LEDMATRIX_Alphabet.txt
22934: 069_pas_LEDMATRIX_Alphabet_run.txt
22935: 069_pas_LEDMATRIX_Alphabet_tester.txt
22936: 069_PAS_LEDMATRIX_COLOR.TXT
22937: 069_pas_ledmatrix_fixedit.txt
22938: 069_pas_LEDMATRIX_soundbox.txt
22939: 069_pas_LEDMATRIX_soundbox2.txt
22940: 069_Richter_MATRIX.TXT
22941: 070_pas_functionplot.txt
22942: 070_pas_functionplotter2.txt
22943: 070_pas_functionplotter2_mx4.txt
22944: 070_pas_functionplotter2_tester.txt
22945: 070_pas_functionplotter3.txt
22946: 070_pas_functionplotter4.txt
22947: 070_pas_functionplotter_digital.txt
22948: 070_pas_functionplotter_elliptic.txt
22949: 070_pas_function_helmholtz.txt
22950: 070_pas_textcheck_experimental.txt
22951: 071_pas_graphics.txt
22952: 071_pas_graphics_drawsym.txt
22953: 071_pas_graphics_drawsym_save.txt
22954: 071_pas_graphics_random.txt
22955: 072_pas_fractals.txt
22956: 072_pas_fractals_2.txt
22957: 072_pas_fractals_blackhole.txt
22958: 072_pas_fractals_perfomance.txt
22959: 072_pas_fractals_perfomance_new.txt
22960: 072_pas_fractals_perfomance_sharp.txt
22961: 072_pas_fractals_performance.txt
22962: 072_pas_fractals_performance_mX4.txt
22963: 073_pas_forms.txt
22964: 074_pas_chartgenerator.txt
22965: 074_pas_chartgenerator_solution.txt
22966: 074_pas_chartgenerator_solution_back.txt
22967: 074_pas_charts.txt
22968: 075_bitmap_Artwork2.txt
22969: 075_pas_bitmappuzzle.txt
22970: 075_pas_bitmappuzzle24.prod.txt
22971: 075_pas_bitmappuzzle2_prod.txt
22972: 075_pas_bitmappuzzle3.txt
22973: 075_pas_bitmapsolve.txt
22974: 075_pas_bitmap_Artwork.txt
22975: 075_pas_puzzlepas_solution.txt
22976: 076_pas_3dcube.txt
22977: 076_pas_circle.txt
22978: 077_pas_mmshow.txt
22979: 078_pas_pi.txt
22980: 079_pas_3dcube_animation.txt
22981: 079_pas_3dcube_animation4.txt
22982: 079_pas_3dcube_plus.txt
22983: 080_pas_hanoi.txt
22984: 080_pas_hanoi2.txt
22985: 080_pas_hanoi2_file.txt
22986: 080_pas_hanoi2_sol.txt
22987: 080_pas_hanoi2_tester.txt
22988: 080_pas_hanoi2_tester_fast.txt
22989: 080_pas_hanoi3.txt
22990: 081_pas_chartist2.txt
22991: 082_pas_biorythmus.txt
22992: 082_pas_biorythmus_solution.txt
22993: 082_pas_biorythmus_solution_3.txt
22994: 082_pas_biorythmus_test.txt
22995: 083_pas_GITARRE.txt
22996: 083_pas_soundbox_tones.txt
22997: 084_pas_waves.txt
22998: 085_mxsinus_logo.txt
22999: 085_sinus_plot_waves.txt
23000: 086_pas_graph_arrow_heart.txt
23001: 087_bitmap_loader.txt
23002: 087_pas_bitmap_solution.txt
23003: 087_pas_bitmap_solution2.txt
23004: 087_pas_bitmap_subimage.txt
23005: 087_pas_bitmap_test.txt
23006: 088_pas_soundbox2_mp3.txt
23007: 088_pas_soundbox_mp3.txt
23008: 088_pas_sphere_2.txt
23009: 089_pas_gradient.txt
23010: 089_pas_maxland2.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
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

23011: 090_pas_sudoku4.txt
23012: 090_pas_sudoku4_2.txt
23013: 091_pas_cube4.txt
23014: 092_pas_statistics4.txt
23015: 093_variance.txt
23016: 093_variance_debug.txt
23017: 094_pas_daysold.txt
23018: 094_pas_stat_date.txt
23019: 095_pas_ki_simulation.txt
23020: 096_pas_geisen_problem.txt
23021: 096_pas_montyhall_problem.txt
23022: 097_lotto_proofofconcept.txt
23023: 097_pas_lottocombinations_beat_plus.txt
23024: 097_pas_lottocombinations_beat_plus2.txt
23025: 097_pas_lottocombinations_universal.txt
23026: 097_pas_lottosimulation.txt
23027: 098_pas_chartgenerator_plus.txt
23028: 099_pas_3D_show.txt
23029: 200_big_numbers.txt
23030: 200_big_numbers2.txt
23031: 201_streamload_xml.txt
23032: 202_systemcheck.txt
23033: 203_webservice_simple_intftester.txt
23034: 204_webservice_simple.txt
23035: 205_future_value_service.txt
23036: 206_DTD_string_functions.txt
23037: 207_ibz2_async_process.txt
23038: 208_crc32_hash.txt
23039: 209_cryptohash.txt
23040: 210_public_private.txt
23041: 210_public_private_cryptosystem.txt
23042: 211_wipe_pattern.txt
23043: 211_wipe_pattern2.txt
23044: 211_wipe_pattern_solution.txt
23045: 212_pas_statisticmodule4.TXT
23046: 212_pas_statisticmoduletxt.TXT
23047: 212_statisticmodule4.txt
23048: 213_pas_BBP_Algo.txt
23049: 214_mxdocudemo.txt
23050: 214_mxdocudemo2.txt
23051: 214_mxdocudemo3.txt
23052: 215_hints_test.TXT
23053: 216_warnings_test.TXT
23054: 217_pas_heartbeat.txt
23055: 218_biorythmus_studio.txt
23056: 219_cipherbox.txt
23057: 219_crypt_source_comtest_solution.TXT
23058: 220_cipherbox_form.txt
23059: 220_cipherbox_form2.txt
23060: 221_bcd_explain.txt
23061: 222_memoform.txt
23062: 223_directorybox.txt
23063: 224_dialogs.txt
23064: 225_sprite_animation.txt
23065: 226_ASCII_Grid2.TXT
23066: 227_animation.txt
23067: 227_animation2.txt
23068: 228_android_calendar.txt
23069: 229_android_game.txt
23070: 229_android_game_tester.txt
23071: 230_DataProvider.txt
23072: 230_DataSetProvider.txt
23073: 230_DataSetXMLBackupScholz.txt
23074: 231_DBGrid_access.txt
23075: 231_DBGrid_XMLaccess.txt
23076: 231_DBGrid_XMLaccess2.txt
23077: 231_DBGrid_XMLaccess_locatetester.txt
23078: 231_DBGrid_XML_CDS_local.txt
23079: 232_outline.txt
23080: 232_outline_2.txt
23081: 233_modular_form.txt
23082: 234_debugoutform.txt
23083: 235_fastform.TXT
23084: 236_componentpower.txt
23085: 236_componentpower_back.txt
23086: 237_pas_4forms.txt
23087: 238_lottogen_form.txt
23088: 239_pas_sierpinski.txt
23089: 239_pas_sierpinski2.txt
23090: 240_unitGlobal_tester.txt
23091: 241_db3_sql_tutorial.txt
23092: 241_db3_sql_tutorial2.txt
23093: 241_db3_sql_tutorial2fix.txt
23094: 241_db3_sql_tutorial3.txt
23095: 241_db3_sql_tutorial3connect.txt
23096: 241_db3_sql_tutorial3_fptest.txt
23097: 241_RTL_SET2.txt
23098: 241_RTL_SET2_tester.txt
23099: 242_Component_Control.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
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

23100: 243_tutorial_loader.txt
23101: 244_script_loader_loop.txt
23102: 245_formapp2.txt
23103: 245_formapp2_tester.txt
23104: 245_formapp2_testerX.txt
23105: 246_httpapp.txt
23106: 247_datecalendar.txt
23107: 248_ASCII_Grid2_sorted.TXT
23108: 249_picture_grid.TXT
23109: 250_tipsandtricks2.txt
23110: 250_tipsandtricks3.txt
23111: 250_tipsandtricks3api.txt
23112: 250_tipsandtricks3_admin_elevation.txt
23113: 250_tipsandtricks3_tester.txt
23114: 250_tipsandtricks4_tester.txt
23115: 250_tipsandtricks4_tester2.txt
23116: 251_compare_noise_gauss.txt
23117: 251_whitenoise.txt
23118: 251_whitenoise2.txt
23119: 252_hilbert_turtle.txt
23120: 252_pas_hilbert.txt
23121: 253_opearatingsystem3.txt
23122: 254_dynarrays.txt
23123: 255_einstein.txt
23124: 256_findconsts_of_EXE.txt
23125: 256_findfunctions2_of_EXE.txt
23126: 256_findfunctions2_of_EXEaverp.txt
23127: 256_findfunctions2_of_EXEspec.txt
23128: 256_findfunctions3.txt
23129: 256_findfunctions_of_EXE.txt
23130: 257_AES_Cipher.txt
23131: 258_AES_cryptobox.txt
23132: 258_AES_cryptobox2.txt
23133: 258_AES_cryptobox2_passdlg.txt
23134: 259_AES_crypt_directory.txt
23135: 260_sendmessage_2.TXT
23136: 260_sendmessage_beta.TXT
23137: 261_probability.txt
23138: 262_mxoutputdemo4.txt
23139: 263_async_sound.txt
23140: 264_vclutils.txt
23141: 264_VCL_utils2.txt
23142: 265_timer_API.txt
23143: 266_serial_interface.txt
23144: 266_serial_interface2.txt
23145: 266_serial_interface3.txt
23146: 267_ackermann_rec.txt
23147: 267_ackermann_variants.txt
23148: 268_DBGrid_tree.txt
23149: 269_record_grid.TXT
23150: 270_Jedi_FunctionPower.txt
23151: 270_Jedi_FunctionPowertester.txt
23152: 271_closures_study.txt
23153: 271_closures_study_workingset2.txt
23154: 272_pas_function_show.txt
23155: 273_pas_function_show2.txt
23156: 274_library_functions.txt
23157: 275_turtle_language.txt
23158: 275_turtle_language_save.txt
23159: 276_save_algo.txt
23160: 276_save_algo2.txt
23161: 277_functionsfor39.txt
23162: 278_DB_Dialogs.TXT
23163: 279_hexer2.TXT
23164: 279_hexer2macro.TXT
23165: 279_hexer2macroback.TXT
23166: 280_UML_process.txt
23167: 280_UML_process_knabe2.txt
23168: 280_UML_process_knabe3.txt
23169: 280_UML_process_TIM_Botzenhardt.txt
23170: 280_UML_TIM_Seitz.txt
23171: 281_picturepuzzle.txt
23172: 281_picturepuzzle2.txt
23173: 281_picturepuzzle3.txt
23174: 281_picturepuzzle4.txt
23175: 479_inputquery.txt
23176: 480_regex_pathfinder2.txt
23177: 482 processPipe.txt
23178: 483_PathFuncTest_mX.txt
23179: 485_InnoFunc.txt
23180: 487_asyncKeyState.txt
23181: 489_simpleComport.txt
23182: 491_analogmeter.txt
23183: 493_gadgets.txt
23184: 496_InstallX.txt
23185: 498_UnitTesting.txt
23186: 500_diceoflifes.txt
23187: 502_findalldocs.txt
23188: 504_fileclass.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
461_sqlform_calwin.txt
462_caesarcipher.txt
463_global_exception.txt
464_function_procedure.txt
464_function_procedure2.txt
464_function_procedure3.txt
465_U_HexView.txt
466_moon.txt
466_moon_inputquery.txt
4671_cardmagic.txt
467_helmholtz_graphic.txt
468_URLMon.txt
468_URLMon2.txt
469_formarrow.txt
469_formarrow_datepicker.txt
469_formarrow_datepicker_ibz_result.txt
469_ibzresult.txt
470_DFFUtils_compiled.txt
470_DFFUtils_ScrollingLED.txt
470_Oscilloscope.txt
470_Oscilloscope_code.txt
471_cardmagic.txt
471_cardmagic2.txt
472_allcards.TXT
473_comboset.txt
474_wakeonlan.txt
474_wakeonlan2.txt
476_getscripttest.txt
477_filenameonly.txt
480_regex_pathfinder.txt
481_processList.txt
482 processPipeGCC.txt
484_filefinder3.txt
486_VideoGrabber.txt
488_asyncTerminal.txt
490_webCamproc.txt
492_snowflake2.txt
495_fourierfreq.txt
497_LED.txt
499_mulu42.txt
501_firebird_datasnap_tests.txt
503_led_switch.txt
505_debug.txt

```
23189: 506_colormatrix.txt                          507_derutils.txt
23190: 508_simplecomportmorse.txt                   509_GEOMap2.txt
23191: 509_509_GEOMap2_SReverse.TXT                 510_510_bonn_gpsdata_mX4.pas
23192: 511_LEDLabel.txt                             512_LED_moon.txt
23193: 513_StreamIntegration.txt                    514_LED_moon2.txt
23194: 515_ledclock3.txt                            516_mapview.txt
23195: 517_animation7.txt                           518_sensors_meter.txt
23196: 519_powtils.txt                              520_run_bytecode.txt
23197: 521_iputils2.txt                             522_getgeocode.txt
23198: 523_NMEA.txt                                 524_NAV_Utils.txt
23199: 525_GEO84s.txt                              526_Compass_meter.txt
23200: 527_GPSDemo.txt                             528_linescount.txt
23201: 529_profilertest.txt                         530_3DLab.txt
23202:
23203:
23204: Web Script Examples:
23205:
23206: http://www.softwareschule.ch/examples/performer.txt';
23207: http://www.softwareschule.ch/examples/turtle.txt';
23208: http://www.softwareschule.ch/examples/SQLExport.txt';
23209: http://www.softwareschule.ch/examples/Richter.txt';
23210: http://www.softwareschule.ch/examples/checker.txt';
23211: http://www.softwareschule.ch/examples/demoscript.txt';
23212: http://www.softwareschule.ch/examples/ibzresult.txt';
23213: http://www.softwareschule.ch/examples/performindex.txt
23214: http://www.softwareschule.ch/examples/processlist.txt
23215: http://www.softwareschule.ch/examples/game.txt
23216: http://www.softwareschule.ch/examples/GEOGPS.txt
23217: http://www.softwareschule.ch/examples/turtle2.txt
23218:
23219:
23220:
23221: Delphi Basics Run Time Library listing
23222: ********************************************************************************
23223: A
23224: Compiler Directive $A Determines whether data is aligned or packed
23225: Compiler Directive $Align Determines whether data is aligned or packed
23226: Compiler Directive $AppType Determines the application type : GUI or Console
23227: Procedure SysUtils Abort Aborts the current processing with a silent exception
23228: Function System Abs Gives the absolute value of a number (-ve sign is removed)
23229: Directive Abstract Defines a class method only implemented in subclasses
23230: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
23231: Function System Addr Gives the address of a variable, function or procedure
23232: Keyword And Boolean and or bitwise and of two arguments
23233: Type System AnsiChar A character type guaranteed to be 8 bits in size
23234: Function SysUtils AnsiCompareStr Compare two strings for equality
23235: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
23236: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
23237: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
23238: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
23239: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
23240: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
23241: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
23242: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
23243: Function StrUtils AnsiPos Find the position of one string in another
23244: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
23245: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
23246: Function StrUtils AnsiRightStr Extracts characters from the right of a string
23247: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
23248: Type System AnsiString A data type that holds a string of AnsiChars
23249: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
23250: Procedure System Append Open a text file to allow appending of text to the end
23251: Procedure SysUtils AppendStr Concatenate one string onto the end of another
23252: Function Math ArcCos The Arc Cosine of a number, returned in radians
23253: Function Math ArcSin The Arc Sine of a number, returned in radians
23254: Function System ArcTan The Arc Tangent of a number, returned in radians
23255: Keyword Array A data type holding indexable collections of data
23256: Keyword As Used for casting object references
23257: Procedure System Assign Assigns a file handle to a binary or text file
23258: Function System Assigned Returns true if a reference is not nil
23259: Procedure System AssignFile Assigns a file handle to a binary or text file
23260: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
23261:
23262: B
23263: Compiler Directive $B Whether to short cut and and or operations
23264: Compiler Directive $BoolEval Whether to short cut and and or operations
23265: Procedure SysUtils Beep Make a beep sound
23266: Keyword Begin Keyword that starts a statement block
23267: Function System BeginThread Begins a separate thread of code execution
23268: Procedure System BlockRead Reads a block of data records from an untyped binary file
23269: Procedure System BlockWrite Writes a block of data records to an untyped binary file
23270: Type System Boolean Allows just True and False values
23271: Function Classes Bounds Create a TRect value from top left and size values
23272: Procedure System Break Forces a jump out of a single loop
23273: Type System Byte An integer type supporting values 0 to 255
23274:
23275: C
23276: Type System Cardinal The basic unsigned integer type
23277: Keyword Case A mechanism for acting upon different values of an Ordinal
```

```
23278: Function StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
23279: Function SysUtils ChangeFileExt Change the extension part of a file name
23280: Type System Char Variable type holding a single character
23281: Procedure System ChDir Change the working drive plus path for a specified drive
23282: Function System Chr Convert an integer into a character
23283: Keyword Class Starts the declaration of a type of object class
23284: Procedure System Close Closes an open file
23285: Procedure System CloseFile Closes an open file
23286: Variable System CmdLine Holds the execution text used to start the current program
23287: Type System Comp A 64 bit signed integer
23288: Function SysUtils CompareStr Compare two strings to see which is greater than the other
23289: Function SysUtils CompareText Compare two strings for equality, ignoring case
23290: Function Math CompareValue Compare numeric values with a tolerance
23291: Function System Concat Concatenates one or more strings into one string
23292: Keyword Const Starts the definition of fixed data values
23293: Keyword Constructor Defines the method used to create an object from a class
23294: Procedure System Continue Forces a jump to the next iteration of a loop
23295: Function ConvUtils Convert Convert one measurement value to another
23296: Function System Copy Create a copy of part of a string or an array
23297: Function System Cos The Cosine of a number
23298: Function SysUtils CreateDir Create a directory
23299: Type System Currency A floating point type with 4 decimals used for financial values
23300: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
23301: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
23302: Variable SysUtils CurrencyString The currency string used in currency display functions
23303: Function SysUtils CurrToStr Convert a currency value to a string
23304: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
23305:
23306: D
23307: Compiler Directive $D Determines whether application debug information is built
23308: Compiler Directive $DebugInfo Determines whether application debug information is built
23309: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
23310: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
23311: Function SysUtils Date Gives the current date
23312: Variable SysUtils DateSeparator The character used to separate display date fields
23313: Function SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
23314: Function SysUtils DateTimeToStr Converts TDateTime date and time values to a string
23315: Procedure SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
23316: Function SysUtils DateToStr Converts a TDateTime date value to a string
23317: Function DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
23318: Function DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
23319: Function DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
23320: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
23321: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
23322: Function DateUtils DaysInAMonth Gives the number of days in a month
23323: Function DateUtils DaysInAYear Gives the number of days in a year
23324: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
23325: Procedure System Dec Decrement an ordinal variable
23326: Variable SysUtils DecimalSeparator The character used to display the decimal point
23327: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
23328: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
23329: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
23330: Directive Default Defines default processing for a property
23331: Function Math DegToRad Convert a degrees value to radians
23332: Procedure System Delete Delete a section of characters from a string
23333: Function SysUtils DeleteFile Delete a file specified by its file name
23334: Keyword Destructor Defines the method used to destroy an object
23335: Function SysUtils DirectoryExists Returns true if the given directory exists
23336: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
23337: Function SysUtils DiskSize Gives the size in bytes of a specified drive
23338: Procedure System Dispose Dispose of storage used by a pointer type variable
23339: Keyword Div Performs integer division, discarding the remainder
23340: Keyword Do Defines the start of some controlled action
23341: Type System Double A floating point type supporting about 15 digits of precision
23342: Keyword DownTo Prefixes an decremental for loop target value
23343: Function StrUtils DupeString Creates a string containing copies of a substring
23344: Directive Dynamic Allows a class method to be overriden in derived classes
23345:
23346: E
23347: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
23348: Compiler Directive $EndIf Terminates conditional code compilation
23349: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
23350: Keyword Else Starts false section of if, case and try statements
23351: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
23352: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
23353: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
23354: Keyword End Keyword that terminates statement blocks
23355: Function DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
23356: Function DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
23357: Procedure System EndThread Terminates a thread with an exit code
23358: Function System Eof Returns true if a file opened with Reset is at the end
23359: Function System Eoln Returns true if the current text file is pointing at a line end
23360: Procedure System Erase Erase a file
23361: Variable System ErrorAddr Sets the error address when an application terminates
23362: Keyword Except Starts the error trapping clause of a Try statement
23363: Procedure System Exclude Exclude a value in a set variable
23364: Procedure System Exit Exit abruptly from a function or procedure
23365: Variable System ExitCode Sets the return code when an application terminates
23366: Function System Exp Gives the exponent of a number
```

```
23367: Directive System Export Makes a function or procedure in a DLL externally available
23368: Type System Extended The floating point type with the highest capacity and precision
23369: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
23370: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
23371: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
23372: Function SysUtils ExtractFileName Extracts the name part of a full file name
23373: Function SysUtils ExtractFilePath Extracts the path part of a full file name
23374:
23375: F
23376: Function StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
23377: Keyword File Defines a typed or untyped file
23378: Function SysUtils FileAge Get the last modified date/time of a file without opening it
23379: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
23380: Function SysUtils FileExists Returns true if the given file exists
23381: Function SysUtils FileGetAttr Gets the attributes of a file
23382: Variable System FileMode Defines how Reset opens a binary file
23383: Function System FilePos Gives the file position in a binary or text file
23384: Function SysUtils FileSearch Search for a file in one or more directories
23385: Function SysUtils FileSetAttr Sets the attributes of a file
23386: Function SysUtils FileSetDate Set the last modified date and time of a file
23387: Function System FileSize Gives the size in records of an open file
23388: Procedure System FillChar Fills out a section of storage with a fill character or byte value
23389: Keyword Finally Starts the unconditional code section of a Try statement
23390: Function SysUtils FindClose Closes a successful FindFirst file search
23391: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
23392: Function SysUtils FindFirst Finds all files matching a file mask and attributes
23393: Function SysUtils FindNext Find the next file after a successful FindFirst
23394: Function SysUtils FloatToStr Convert a floating point value to a string
23395: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
23396: Procedure System Flush Flushes buffered text file data to the file
23397: Keyword For Starts a loop that executes a finite number of times
23398: Function SysUtils ForceDirectories Create a new path of directories
23399: Function SysUtils Format Rich formatting of numbers and text into a string
23400: Function SysUtils FormatCurr Rich formatting of a currency value into a string
23401: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
23402: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
23403: Function System Frac The fractional part of a floating point number
23404: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
23405: Procedure System FreeMem Free memory storage used by a variable
23406: Keyword System Function Defines a subroutine that returns a value
23407:
23408: G
23409: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
23410: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
23411: Function System GetLastError Gives the error code of the last failing Windows API call
23412: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
23413: Function System GetMem Get a specified number of storage bytes
23414: Keyword Goto Forces a jump to a label, regardless of nesting
23415:
23416: H
23417: Compiler Directive $H Treat string types as AnsiString or ShortString
23418: Compiler Directive $Hints Determines whether Delphi shows compilation hints
23419: Procedure System Halt Terminates the program with an optional dialog
23420: Function System Hi Returns the hi-order byte of a (2 byte) Integer
23421: Function System High Returns the highest value of a type or variable
23422:
23423: I
23424: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
23425: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
23426: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
23427: Compiler Directive $IfOpt Tests for the state of a Compiler directive
23428: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
23429: Compiler Directive $IOChecks When on, an IO operation error throws an exception
23430: Keyword If Starts a conditional expression to determine what to do next
23431: Keyword Implementation Starts the implementation (code) section of a Unit
23432: Keyword In Used to test if a value is a member of a set
23433: Procedure System Inc Increment an ordinal variable
23434: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
23435: Procedure System Include Include a value in a set variable
23436: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
23437: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
23438: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
23439: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
23440: Function DateUtils IncYear Increments a TDateTime variable by a number of years
23441: Directive Index Principally defines indexed class data properties
23442: Constant Math Infinity Floating point value of infinite size
23443: Keyword Inherited Used to call the parent class constructor or destructor method
23444: Variable System Input Defines the standard input text file
23445: Function Dialogs InputBox Display a dialog that asks for user text input, with default
23446: Function Dialogs InputQuery Display a dialog that asks for user text input
23447: Procedure System Insert Insert a string into another string
23448: Function System Int The integer part of a floating point number as a float
23449: Type System Int64 A 64 bit sized integer - the largest in Delphi
23450: Type System Integer The basic Integer type
23451: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
23452: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
23453: Function SysUtils IntToStr Convert an integer into a string
23454: Function System IOResult Holds the return code of the last I/O operation
23455: Keyword Is Tests whether an object is a certain class or ascendant
```

```
23456: Function Math IsInfinite Checks whether a floating point number is infinite
23457: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
23458: Function System IsMultiThread Returns true if the code is running multiple threads
23459: Function Math IsNaN Checks to see if a floating point number holds a real number
23460:
23461: L
23462: Compiler Directive $L Determines what application debug information is built
23463: Compiler Directive $LocalSymbols Determines what application debug information is built
23464: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
23465: Function SysUtils LastDelimiter Find the last position of selected characters in a string
23466: Function System Length Return the number of elements in an array or string
23467: Function System Ln Gives the natural logarithm of a number
23468: Function System Lo Returns the low-order byte of a (2 byte) Integer
23469: Function Math Log10 Gives the log to base 10 of a number
23470: Variable SysUtils LongDateFormat Long version of the date to string format
23471: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
23472: Type System LongInt An Integer whose size is guaranteed to be 32 bits
23473: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
23474: Variable SysUtils LongTimeFormat Long version of the time to string format
23475: Type System LongWord A 32 bit unsigned integer
23476: Function System Low Returns the lowest value of a type or variable
23477: Function SysUtils LowerCase Change upper case characters in a string to lower case
23478:
23479: M
23480: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
23481: Function Math Max Gives the maximum of two integer values
23482: Constant System MaxInt The maximum value an Integer can have
23483: Constant System MaxLongInt The maximum value an LongInt can have
23484: Function Math Mean Gives the average for a set of numbers
23485: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
23486: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
23487: Function Math Min Gives the minimum of two integer values
23488: Constant SysUtils MinsPerDay Gives the number of minutes in a day
23489: Procedure System MkDir Make a directory
23490: Keyword Mod Performs integer division, returning the remainder
23491: Constant SysUtils MonthDays Gives the number of days in a month
23492: Function DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
23493: Procedure System Move Copy bytes of data from a source to a destination
23494:
23495: N
23496: Constant Math NaN Not a real number
23497: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
23498: Procedure System New Create a new pointer type variable
23499: Constant System Nil A pointer value that is defined as undetermined
23500: Keyword Not Boolean Not or bitwise not of one arguments
23501: Function SysUtils Now Gives the current date and time
23502: Variable Variants Null A variable that has no value
23503:
23504: O
23505: Compiler Directive $O Determines whether Delphi optimises code when compiling
23506: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
23507: Compiler Directive $OverFlowChecks Determines whether Delphi checks integer and enum bounds
23508: Keyword System Object Allows a subroutine data type to refer to an object method
23509: Function System Odd Tests whether an integer has an odd value
23510: Keyword Of Linking keyword used in many places
23511: Keyword On Defines exception handling in a Try Except clause
23512: Keyword Or Boolean or or bitwise or of two arguments
23513: Function System Ord Provides the Ordinal value of an integer, character or enum
23514: Directive Out Identifies a routine parameter for output only
23515: Variable System Output Defines the standard output text file
23516: Directive Overload Allows 2 or more routines to have the same name
23517: Directive Override Defines a method that replaces a virtual parent class method
23518:
23519: P
23520: Keyword Packed Compacts complex data types into minimal storage
23521: Type System PAnsiChar A pointer to an AnsiChar value
23522: Type System PAnsiString Pointer to an AnsiString value
23523: Function System ParamCount Gives the number of parameters passed to the current program
23524: Function System ParamStr Returns one of the parameters used to run the current program
23525: Type System PChar A pointer to an Char value
23526: Type System PCurrency Pointer to a Currency value
23527: Type System PDateTime Pointer to a TDateTime value
23528: Type System PExtended Pointer to a Extended floating point value
23529: Function System Pi The mathematical constant
23530: Type System PInt64 Pointer to an Int64 value
23531: Function Classes Point Generates a TPoint value from X and Y values
23532: Type System Pointer Defines a general use Pointer to any memory based data
23533: Function Classes PointsEqual Compares two TPoint values for equality
23534: Function System Pos Find the position of one string in another
23535: Function System Pred Decrement an ordinal variable
23536: Function Printers Printer Returns a reference to the global Printer object
23537: Directive Private Starts the section of private data and methods in a class
23538: Keyword System Procedure Defines a subroutine that does not return a value
23539: Procedure FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
23540: Keyword System Program Defines the start of an application
23541: Function Dialogs PromptForFileName Shows a dialog allowing the user to select a file
23542: Keyword System Property Defines controlled access to class fields
23543: Directive Protected Starts a section of class private data accesible to sub-classes
23544: Type System PShortString A pointer to an ShortString value
```

```
23545: Type System PString Pointer to a String value
23546: Function Types PtInRect Tests to see if a point lies within a rectangle
23547: Directive Public Starts an externally accessible section of a class
23548: Directive Published Starts a published externally accessible section of a class
23549: Type System PVariant Pointer to a Variant value
23550: Type System PWideChar Pointer to a WideChar
23551: Type System PWideString Pointer to a WideString value
23552:
23553: Q
23554: Compiler Directive $Q Determines whether Delphi checks integer and enum bounds
23555:
23556: R
23557: Compiler Directive $R Determines whether Delphi checks array bounds
23558: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
23559: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
23560: Compiler Directive $Resource Defines a resource file to be included in the application linking
23561: Function Math RadToDeg Converts a radian value to degrees
23562: Keyword Raise Raise an exception
23563: Function System Random Generate a random floating point or integer number
23564: Procedure System Randomize Reposition the Random number generator next value
23565: Function Math RandomRange Generate a random integer number within a supplied range
23566: Variable System RandSeed Reposition the Random number generator next value
23567: Procedure System Read Read data from a binary or text file
23568: Procedure System ReadLn Read a complete line of data from a text file
23569: Type System Real A floating point type supporting about 15 digits of precision
23570: Type System Real48 The floating point type with the highest capacity and precision
23571: Procedure System ReallocMem Reallocate an existing block of storage
23572: Function DateUtils RecodeDate Change only the date part of a TDateTime variable
23573: Function DateUtils RecodeTime Change only the time part of a TDateTime variable
23574: Keyword Record A structured data type - holding fields of data
23575: Function Classes Rect Create a TRect value from 2 points or 4 coordinates
23576: Function SysUtils RemoveDir Remove a directory
23577: Procedure System Rename Rename a file
23578: Function SysUtils RenameFile Rename a file or directory
23579: Keyword Repeat Repeat statements until a ternmination condition is met
23580: Procedure SysUtils ReplaceDate Change only the date part of a TDateTime variable
23581: Procedure SysUtils ReplaceTime Change only the time part of a TDateTime variable
23582: Procedure System Reset Open a text file for reading, or binary file for read/write
23583: Variable System Result A variable used to hold the return value from a function
23584: Procedure System ReWrite Open a text or binary file for write access
23585: Procedure System RmDir Remove a directory
23586: Function System Round Rounds a floating point number to an integer
23587: Procedure System RunError Terminates the program with an error dialog
23588:
23589: S
23590: Constant SysUtils SecsPerDay Gives the number of seconds in a day
23591: Procedure System Seek Move the pointer in a binary file to a new record position
23592: Function System SeekEof Skip to the end of the current line or file
23593: Function System SeekEoln Skip to the end of the current line or file
23594: Function FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
23595: Variable System Self Hidden parameter to a method - refers to the containing object
23596: Keyword Set Defines a set of up to 255 distinct values
23597: Function SysUtils SetCurrentDir Change the current directory
23598: Procedure System SetLength Changes the size of a string, or the size(s) of an array
23599: Procedure System SetString Copies characters from a buffer into a string
23600: Keyword Shl Shift an integer value left by a number of bits
23601: Variable SysUtils ShortDateFormat Compact version of the date to string format
23602: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
23603: Type System ShortInt An integer type supporting values -128 to 127
23604: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
23605: Type System ShortString Defines a string of up to 255 characters
23606: Variable SysUtils ShortTimeFormat Short version of the time to string format
23607: Procedure Dialogs ShowMessage Display a string in a simple dialog with an OK button
23608: Procedure Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
23609: Procedure Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
23610: Keyword Shr Shift an integer value right by a number of bits
23611: Function System Sin The Sine of a number
23612: Type System Single The smallest capacity and precision floating point type
23613: Function System SizeOf Gives the storage byte size of a type or variable
23614: Function System Slice Creates a slice of an array as an Open Array parameter
23615: Type System SmallInt An Integer type supporting values from -32768 to 32767
23616: Function System Sqr Gives the square of a number
23617: Function System Sqrt Gives the square root of a number
23618: Procedure System Str Converts an integer or floating point number to a string
23619: Type System String A data type that holds a string of characters
23620: Function System StringOfChar Creates a string with one character repeated many times
23621: Function SysUtils StringReplace Replace one or more substrings found within a string
23622: Function System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
23623: Function SysUtils StrScan Searches for a specific character in a constant string
23624: Function SysUtils StrToCurr Convert a number string into a currency value
23625: Function SysUtils StrToDate Converts a date string into a TDateTime value
23626: Function SysUtils StrToDateTime Converts a date+time string into a TDateTime value
23627: Function SysUtils StrToFloat Convert a number string into a floating point value
23628: Function SysUtils StrToInt Convert an integer string into an Integer value
23629: Function SysUtils StrToInt64 Convert an integer string into an Int64 value
23630: Function SysUtils StrToInt64Def Convert a string into an Int64 value with default
23631: Function SysUtils StrToIntDef Convert a string into an Integer value with default
23632: Function SysUtils StrToTime Converts a time string into a TDateTime value
23633: Function StrUtils StuffString Replaces a part of one string with another
```

```
23634: Function System Succ Increment an ordinal variable
23635: Function Math Sum Return the sum of an array of floating point values
23636:
23637: T
23638: Function Math Tan The Tangent of a number
23639: Type Classes TBits An object that can hold an infinite number of Boolean values
23640: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
23641: Type ConvUtils TConvType Defines a measurement type as used by Convert
23642: Type System TDateTime Data type holding a date and time value
23643: Type System Text Defines a file as a text file
23644: Type System TextFile Declares a file type for storing lines of text
23645: Type SysUtils TFloatFormat Formats for use in floating point number display functions
23646: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
23647: Keyword Then Part of an if statement - starts the true clause
23648: Variable SysUtils ThousandSeparator The character used to display the thousands separator
23649: Keyword ThreadVar Defines variables that are given separate instances per thread
23650: Function SysUtils Time Gives the current time
23651: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
23652: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
23653: Variable SysUtils TimeSeparator The character used to separate display time fields
23654: Function SysUtils TimeToStr Converts a TDateTime time value to a string
23655: Type Classes TList General purpose container of a list of objects
23656: Keyword To Prefixes an incremental for loop target value
23657: Type System TObject The base class type that is ancestor to all other classes
23658: Function DateUtils Tomorrow Gives the date tomorrow
23659: Type Dialogs TOpenDialog Displays a file selection dialog
23660: Type Types TPoint Holds X and Y integer values
23661: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
23662: Type Types TRect Holds rectangle coordinate values
23663: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
23664: Function SysUtils Trim Removes leading and trailing blanks from a string
23665: Function SysUtils TrimLeft Removes leading blanks from a string
23666: Function SysUtils TrimRight Removes trailing blanks from a string
23667: Function System Trunc The integer part of a floating point number
23668: Procedure System Truncate Truncates a file size - removes all data after the current position
23669: Keyword Try Starts code that has error trapping
23670: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
23671: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
23672: Type Classes TStringList Holds a variable length list of strings
23673: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
23674: Type System TThreadFunc Defines the function to be called by BeginThread
23675: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
23676: Keyword Type Defines a new category of variable or process
23677:
23678:
23679: U
23680: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
23681: Keyword Unit Defines the start of a unit file - a Delphi module
23682: Keyword Until Ends a Repeat control loop
23683: Function System UpCase Convert a Char value to upper case
23684: Function SysUtils UpperCase Change lower case characters in a string to upper case
23685: Keyword Uses Declares a list of Units to be imported
23686:
23687: V
23688: Procedure System Val Converts number strings to integer and floating point values
23689: Keyword Var Starts the definition of a section of data variables
23690: Type System Variant A variable type that can hold changing data types
23691: Function Variants VarType Gives the current type of a Variant variable
23692: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
23693: Directive Virtual Allows a class method to be overriden in derived classes
23694:
23695: W
23696: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
23697: Keyword While Repeat statements whilst a continuation condition is met
23698: Type System WideChar Variable type holding a single International character
23699: Function System WideCharToString Copies a null terminated WideChar string to a normal string
23700: Type System WideString A data type that holds a string of WideChars
23701: Keyword With A means of simplifying references to structured variables
23702: Type System Word An integer type supporting values 0 to 65535
23703: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
23704: Procedure System Write Write data to a binary or text file
23705: Procedure System WriteLn Write a complete line of data to a text file
23706:
23707: X
23708: Compiler Directive $X Controls some Pascal extension handling
23709: Keyword Xor Boolean Xor or bitwise Xor of two arguments
23710:
23711: Y
23712: Compiler Directive $Y Determines whether application symbol information is built
23713: Function DateUtils Yesterday Gives the date yesterday
23714:
23715: Z
23716: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
23717:
23718: mapX:
23719:  if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
23720:     writeln('cologne map found');
23721:     GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
23722:     writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
```

```
23723:     OpenMapX('church trier');
23724:     GetGeoCode(C_form,apath: string; const data: string; sfile: boolean): string;
23725:     writeln(GetGeoCode('xml',ExePath+'outputmap_2cologne.xml','cathedral cologne',false));}
23726:     >>>     //latitude: '50.94133705'  longitude: '6.95812076100766'
23727:     // type TPos = (tLat, tLon);TShowFmt = (sfNautical, sfStatute, sfMetric);
23728:     CoordinateStr(Idx: Integer; PosInSec: Double; PosLn: TPos): string;
23729:
23730:
23731:
23732: maXbox Ref:
23733: Signature:
23734: SHA1:  maXbox3.exe 786FA3035226AC3F62F6F6B9E2E2B94C1D659297
23735: CRC32: maXbox3.exe 612D41BB
23736: Ref:
23737:     1. writeln(SHA1(Exepath+'\maxbox3.exe'))
23738:     2. shdig: TSHA1Digest;
23739:        shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
23740:                for i:= 0 to 19 do write(BytetoHex(shdig[i]));
23741:
23742:     3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
23743:
23744: https://www.virustotal.com/en/file/...../
23745:
23746: MD5 7815ee7c5006c542a7a5788315b3a849
23747: SHA1 786fa3035226ac3f62f6f6b9e2e2b94c1d659297
23748: SHA256 7b277b3cfefaa7463523698ae79831edaa3c434998ef77c90c5a903da01c9777
23749:
23750: File name:  maXbox3.exe
23751:
23752: Copyright Free Pascal Script
23753: Publisher kleiner kommunikation
23754: Product maXbox
23755: Original name maxbox3_9.exe
23756: Internal name Sunprise mX4
23757: File version 3.9.9.101
23758: Description maXbox Delphi VM
23759: Comments reduce to the max
23760:
23761: Target machine Intel 386 or later processors and compatible processors
23762: Compilation timestamp 2014-10-24 12:02:35
23763: Entry Point 0x01188B40
23764: Number of sections 10
23765:
23766: PE sections
23767: Name Virtual address Virtual size Raw size Entropy MD5
23768:
23769: .text 4096 18333480 18333696 6.60 03b09af285de1cab7d3335921ce379a5
23770: .itext 18337792 48260 48640 6.55 36bd0c461c0f0693fcaa9a2690dd66c4
23771: .data 18386944 263080 263168 5.58 953e203486b167223c1c1b1fc6ae5759
23772: .bss 18653184 381088 0 0.00 d41d8cd98f00b204e9800998ecf8427e
23773: .idata 19038208 54746 54784 5.63 a7e4bb62133d7c6fcd399937fcaec7f6
23774: .edata 19095552 77 512 0.90 710d2a7ee4fa0f82106348065b3ff46e
23775: .tls 19099648 208 0 0.00 d41d8cd98f00b204e9800998ecf8427e
23776: .rdata 19103744 24 512 0.28 ef2f75905b778426c522854ce5b1c0a0
23777: .reloc 19107840 1168088 1168384 6.76 db20be0be3f5b5489540674ece1359d9
23778: .rsrc 20279296 3466240 3466240 5.27 ca5f9f6c1ee0a92a3f2b04d1d4c03372
23779:
23780: authentihash 16af80d4ecfa98d43579b17a2c37a6cd0ca620b334377fb747221d6187177465
23781: imphash a0d192a8ed78c25f548447101658dad5
23782: File size 22.3 MB ( 23336960 bytes )
23783: File type Win32 EXE
23784: Magic literal
23785: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
23786:
23787: TrID  Windows ActiveX control (36.4%)
23788: Inno Setup installer (34.3%)
23789: InstallShield setup (13.4%)
23790: Win32 EXE PECompact compressed (generic) (13.0%)
23791: Win32 Executable (generic) (1.4%)
23792:
23793: RT_BITMAP 743
23794: RT_STRING 304
23795: RT_RCDATA 156
23796: RT_ICON 49
23797: RT_GROUP_ICON 43
23798: RT_CURSOR 31
23799: RT_GROUP_CURSOR 26
23800: WAVE 9
23801: RT_DIALOG 2
23802: RT_HTML 2
23803: RT_MESSAGETABLE 1
23804: RT_MANIFEST 1
23805: RT_VERSION 1
23806:
23807:  ExifTool file metadata
23808: SpecialBuild
23809: mX4
23810: CodeSize
23811: 18382336
```

```
23812: SubsystemVersion 4.0
23813: Comments reduce to the max
23814: LinkerVersion 2.25
23815: ImageVersion 0.0
23816: FileSubtype 0
23817: FileVersionNumber 3.9.9.101
23818: LanguageCode German (Swiss)
23819: FileFlagsMask 0x003f
23820: FileDescription maXbox Delphi VM
23821: CharacterSet Windows, Latin1
23822: InitializedDataSize 4953600
23823: FileOS Win32
23824: TimeStamp 2014:10:24 13:02:35+01:00
23825: MIMEType application/octet-stream
23826: LegalCopyright Free Pascal Script
23827: FileVersion 3.9.9.101
23828: SpeziellesBuild mX4 Compiler Engine
23829: FileType Win32 EXE
23830: PEType PE32
23831: InternalName Sunprise mX4
23832: FileAccessDate 2014:10:24 16:54:28+01:00
23833: ProductVersion 3.9 Solar mX4
23834:
23835:  PE imports
23836: [+] AVICAP32.DLL  [+] AVICAP32.dll [+] GLU32.dll
23837: [+] IMAGEHLP.DLL  [+] MSVCRT.DLL   [+] MSVFW32.DLL
23838: [+] OpenGL32.dll  [+] SHFolder.dll [+] URLMON.DLL
23839: [+] advapi32.dll  [+] comctl32.dll [+] comdlg32.dll
23840: [+] gdi32.dll     [+] imagehlp.dll [+] imm32.dll
23841: [+] iphlpapi.dll  [+] kernel32.dll [+] mpr.dll
23842: [+] msacm32.dll   [+] ole32.dll    [+] oleacc.dll
23843: [+] oleaut32.dll  [+] oledlg.dll   [+] opengl32.dll
23844: [+] shell32.dll   [+] shlwapi.dll  [+] user32.dll
23845: [+] usp10.dll     [+] version.dll  [+] winhttp.dll
23846: [+] wininet.dll   [+] winmm.dll    [+] winspool.drv
23847: [+] ws2_32.dll    [+] wsock32.dll
23848:
23849: https://www.virustotal.
       com/en/file/7b277b3cfefaa7463523698ae79831edaa3c434998ef77c90c5a903da01c9777/analysis/1414166062/
23850:   ---- bigbitbox code_cleared_checked----
```