```
 1: ********************************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9
 3: ********************************************************************************
 4:
 5: ////////////////////////////////////////////////////////////////////////////////
 6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX
 7: --------------------------------------------------------------------------------
 8:
 9: File Size of EXE: 21267968 V3.9.9.96 June 2014 To EKON/BASTA/JAX/IBZ/SWS
10: ****************Now the Funclist*****************************************
11: Funclist Function : 12682 //10766 //10165 (7648)
12: ****************Now the Proclist*****************************************
13: Proclist Procedure Size is: 7858  //6792 //6401 4752
14: ****************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1275 //995 //
16: def head:max: maXbox7: 22.05.2014 19:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: --------------------------------------------------------------------------------
20: Funclist total Size all is: 21815! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 21155
22: ASize of EXE: 21267968 (16586240) (13511680) (13023744)
23: SHA1 Hash of maXbox 3.9.9.96: EDD7FC051FF703851938AA33B4FBACCBA2552F1C
24:
25: --------------------------------------------------------------------------------
26: --------------------------------------------------------------------------------
27: ////////////////////////////////////////////////////////////////////////////////
28:
29:
30: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
31: Function *************Now the Funclist*****************
32: function  GetResStringChecked(Ident: string; const Args: array of const): string
33: Function ( Index : Longint) : Integer
34: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
35: Function _CheckAutoResult( ResultCode : HResult) : HResult
36: function _T(Name: tbtString): Variant;
37: function ABNFToText(const AText : String) : String
38: Function Abs(e : Extended) : Extended;
39: Function Ackermann( const A, B : Integer) : Integer
40: Function AcquireLayoutLock : Boolean
41: Function ActionByName( const AName : string) : TWebActionItem
42: Function ACTIVEBUFFER : PCHAR
43: Function Add : TAggregate
44: function Add : TCollectionItem
45: Function Add : TColumn
46: Function Add : TComboExItem
47: Function Add : TCookie
48: Function Add : TCoolBand
49: Function Add : TFavoriteLinkItem
50: Function Add : TFileTypeItem
51: Function Add : THeaderSection
52: Function Add : THTMLTableColumn
53: Function Add : TIdEMailAddressItem
54: Function Add : TIdMessagePart
55: Function Add : TIdUserAccount
56: Function Add : TListColumn
57: Function Add : TListItem
58: Function Add : TStatusPanel
59: Function Add : TTaskDialogBaseButtonItem
60: Function Add : TWebActionItem
61: Function Add : TWorkArea
62: Function Add( AClass : TClass) : Integer
63: Function Add( AComponent : TComponent) : Integer
64: Function Add( AItem, AData : Integer) : Integer
65: Function Add( AItem, AData : Pointer) : Pointer
66: Function Add( AItem, AData : TObject) : TObject
67: Function Add( AObject : TObject) : Integer
68: Function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
69: Function Add( const S : WideString) : Integer
70: Function Add( Image, Mask : TBitmap) : Integer
71: Function Add( Index : LongInt; const Text : string) : LongInt
72: Function Add( Sibling : TTreeNode; const S : string) : TTreeNode
73: Function Add(const S: string): Integer
74: function Add(S: string): Integer;
75: Function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
76: Function ADDCHILD : TFIELDDEF
77: Function AddChild( Index : LongInt; const Text : string) : LongInt
78: Function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
79: Function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
80: Function AddChildObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
81: Function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
82: Function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
83: Function ADDFIELDDEF : TFIELDDEF
84: Function AddFileExtIfNecessary( AFileName, AExt : string) : string
85: Function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
86: Function AddIcon( Image : TIcon) : Integer
87: Function AddImage( Value : TCustomImageList; Index : Integer) : Integer
88: Function ADDINDEXDEF : TINDEXDEF
89: Function AddItem(const Caption:String;const ImageIdx,SelectImageIdx,OverlayImagIdx,
    Indent:Int;Data:Ptr):TComboExItem
```

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

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

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

```
357: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;
358: Function ClearBit5( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
359: Function ClearBit6( const Value : Int64; const Bit : TBitRange) : Int64;
360: function CLIENTTOSCREEN(POINT:TPOINT):TPOINT
361: Function Clipboard : TClipboard
362: Function ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
363: Function ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
364: Function ClipLine( var X1, Y1, X2, Y2 : Integer; const ClipRect : TRect) : Boolean;
365: Function ClipLineToRect( var P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean
366: Function Clone( out stm : IStream) : HResult
367: Function CloneConnection : TSQLConnection
368: Function CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
369: function CLOSEQUERY:BOOLEAN
370: Function CloseVolume( var Volume : THandle) : Boolean
371: Function CloseHandle(Handle: Integer): Integer; stdcall;
372: Function CPlApplet( hwndCPl : THandle; uMsg : DWORD; lParam1, lParam2 : Longint) : Longint
373: Function CmdLine: PChar;
374: function CmdShow: Integer;
375: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
376: Function Color32( WinColor : TColor) : TColor32;
377: Function Color321( const Index : Byte; const Palette : TPalette32) : TColor32;
378: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale :BOOLean) : TColor
379: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
380: Function ColorToHTML( const Color : TColor) : String
381: function ColorToIdent(Color: Longint; var Ident: string): Boolean)
382: Function ColorToRGB(color: TColor): Longint
383: function ColorToString(Color: TColor): string)
384: Function ColorToWebColorName( Color : TColor) : string
385: Function ColorToWebColorStr( Color : TColor) : string
386: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn
387: Function Combination(npr, ncr: integer): extended;
388: Function CombinationInt(npr, ncr: integer): Int64;
389: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
390: Function CommaAdd( const AStr1, AStr2 : String) : string
391: Function CommercialRound( const X : Extended) : Int64
392: Function Commit( grfCommitFlags : Longint) : HResult
393: Function Compare( const NameExt : string) : Boolean
394: function CompareDate(const A, B: TDateTime): TValueRelationship;
395: Function CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Integer
396: Function CompareFiles(const FN1,FN2 :string; Breathe:TNotifyEvent;BreathingSender:TObject): boolean
397: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean
398: Function CompareStr( S1, S2 : string) : Integer
399: function CompareStr(const S1: string; const S2: string): Integer)
400: function CompareString(const S1: string; const S2: string): Integer)
401: Function CompareText( S1, S2 : string) : Integer
402: function CompareText(const S1: string; const S2: string): Integer)
403: Function CompareTextLike(cWildStr,cStr:string;const cWildChar:char;lCaseSensitive:boolean): boolean
404: function CompareTime(const A, B: TDateTime): TValueRelationship;
405: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
406: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
407: Function ComponentTypeToString( const ComponentType : DWORD) : string
408: Function CompToCurrency( Value : Comp) : Currency
409: Function CompToDouble( Value : Comp) : Double
410: function ComputeFileCRC32(const FileName : String) : Integer;
411: function ComputeSHA256(astr: string; amode: char): string)  //mode F:File, S:String
412: function ComputeSHA512(astr: string; amode: char): string)  //mode F:File, S:String
413: Function Concat(s: string): string
414: Function ConnectAndGetAll : string
415: Function Connected : Boolean
416: function constrain(x, a, b: integer): integer;
417: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
418: Function ConstraintsDisabled : Boolean
419: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
420: Function ContainsState( oState : TniRegularExpressionState) : boolean
421: Function ContainsStr( const AText, ASubText : string) : Boolean
422: Function ContainsText( const AText, ASubText : string) : Boolean
423: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean
424: Function Content : string
425: Function ContentFromStream( Stream : TStream) : string
426: Function ContentFromString( const S : string) : string
427: Function CONTROLSDISABLED : BOOLEAN
428: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
429: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
430: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
431: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
432: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
433: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
434: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
435: Function ConvTypeToDescription( const AType : TConvType) : string
436: Function ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
437: Function ConvTypeToFamily( const AType : TConvType) : TConvFamily;
438: Function ConvAdd(const AVal:Double;const AType1:TConvType;const AVal2:Double;const AType2,
     AResultType:TConvType): Double
439: Function ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
     AType2:TConvType): TValueRelationship
440: Function ConvDec( const AValue : Double; const AType, AAmountType : TConvType) : Double;
441: Function ConvDec1(const AValue:Double; const AType: TConvType;const AAmount:Double;const
     AAmountType:TConvType):Double;
442: Function ConvDiff(const AVal1:Double;const AType1:TConvType;const AVal2:Double;const AType2,
     AResuType:TConvType):Double
```

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

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

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

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

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

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

```
963: Function FormatValue( AValue : Cardinal) : string
964: Function FormatVersionString( const HiV, LoV : Word) : string;
965: Function FormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
966: function Frac(X: Extended): Extended);
967: Function FreeResource( ResData : HGLOBAL) : LongBool
968: Function FromCommon( const AValue : Double) : Double
969: function FromCommon(const AValue: Double): Double;
970: Function FTPGMTDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
971: Function FTPLocalDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
972: Function FTPMLSToGMTDateTime( const ATimeStamp : String) : TDateTime
973: Function FTPMLSToLocalDateTime( const ATimeStamp : String) : TDateTime
974: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
975: //Function Funclist Size is: 6444 of mX3.9.8.9
976: Function FutureValue(const Rate:Extended;NPeriods:Integer;const Payment,PresentValue:Extended;PaymentTime:
     TPaymentTime):Extended
977: Function Gauss( const x, Spread : Double) : Double
978: function Gauss(const x,Spread: Double): Double;
979: Function GCD(x, y : LongInt) : LongInt;
980: Function GCDJ( X, Y : Cardinal) : Cardinal
981: Function GDAL: LongWord
982: Function GdiFlush : BOOL
983: Function GdiSetBatchLimit( Limit : DWORD) : DWORD
984: Function GdiGetBatchLimit : DWORD
985: Function GenerateHeader : TIdHeaderList
986: Function GeometricMean( const X : TDynFloatArray) : Float
987: Function Get( AURL : string) : string;
988: Function Get2( AURL : string) : string;
989: Function Get8087CW : Word
990: function GetActiveOleObject(const ClassName: String): IDispatch;
991: Function GetAliasDriverName( const AliasName : string) : string
992: Function GetAPMBatteryFlag : TAPMBatteryFlag
993: Function GetAPMBatteryFullLifeTime : DWORD
994: Function GetAPMBatteryLifePercent : Integer
995: Function GetAPMBatteryLifeTime : DWORD
996: Function GetAPMLineStatus : TAPMLineStatus
997: Function GetAppdataFolder : string
998: Function GetAppDispatcher : TComponent
999: function GetArrayLength: integer;
1000: Function GetASCII: string;
1001: Function GetASCIILine: string;
1002: Function GetAsHandle( Format : Word) : THandle
1003: function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
1004: Function GetBackupFileName( const FileName : string) : string
1005: Function GetBBitmap( Value : TBitmap) : TBitmap
1006: Function GetBIOSCopyright : string
1007: Function GetBIOSDate : TDateTime
1008: Function GetBIOSExtendedInfo : string
1009: Function GetBIOSName : string
1010: Function getBitmap(apath: string): TBitmap;
1011: Function GetBitmap( Index : Integer; Image : TBitmap) : Boolean  //object
1012: Function getBitMapObject(const bitmappath: string): TBitmap;
1013: Function GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
1014: Function GetCapsLockKeyState : Boolean
1015: function GetCaptureControl: TControl;
1016: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char) : TJclCdTrackInfo;
1017: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char) : string;
1018: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) : string
1019: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char) : string
1020: Function GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1021: Function GetClockValue : Int64
1022: function getCmdLine: PChar;
1023: function getCmdShow: Integer;
1024: function GetCPUSpeed: Double;
1025: Function GetColField( DataCol : Integer) : TField
1026: Function GetColorBlue( const Color : TColor) : Byte
1027: Function GetColorFlag( const Color : TColor) : Byte
1028: Function GetColorGreen( const Color : TColor) : Byte
1029: Function GetColorRed( const Color : TColor) : Byte
1030: Function GetComCtlVersion : Integer
1031: Function GetComPorts: TStringlist;
1032: Function GetCommonAppdataFolder : string
1033: Function GetCommonDesktopdirectoryFolder : string
1034: Function GetCommonFavoritesFolder : string
1035: Function GetCommonFilesFolder : string
1036: Function GetCommonProgramsFolder : string
1037: Function GetCommonStartmenuFolder : string
1038: Function GetCommonStartupFolder : string
1039: Function GetComponent( Owner, Parent : TComponent) : TComponent
1040: Function GetConnectionRegistryFile( DesignMode : Boolean) : string
1041: Function GetCookiesFolder : string
1042: Function GetCPUSpeed( var CpuSpeed : TFreqInfo) : Boolean
1043: Function GetCurrent : TFavoriteLinkItem
1044: Function GetCurrent : TListItem
1045: Function GetCurrent : TTaskDialogBaseButtonItem
1046: Function GetCurrent : TToolButton
1047: Function GetCurrent : TTreeNode
1048: Function GetCurrent : WideString
1049: Function GetCurrentDir : string
1050: function GetCurrentDir: string)
```

```
1051: Function GetCurrentFolder : string
1052: Function GETCURRENTRECORD( BUFFER : PCHAR) : BOOLEAN
1053: Function GetCurrentProcessId : TIdPID
1054: Function GetCurrentThreadHandle : THandle
1055: Function GetCurrentThreadID: LongWord; stdcall;
1056: Function GetCustomHeader( const Name : string) : String
1057: Function GetDataItem( Value : Pointer) : Longint
1058: Function GetDataLinkFiles( FileNames : TWideStrings; Directory : string) : Integer;
1059: Function GetDataLinkFiles1( FileNames : TStrings; Directory : string) : Integer;
1060: Function GETDATASIZE : INTEGER
1061: Function GetDC(hdwnd: HWND): HDC;
1062: Function GetDefaultFileExt( const MIMEType : string) : string
1063: Function GetDefaults : Boolean
1064: Function GetDefaultSchemaName : WideString
1065: Function GetDefaultStreamLoader : IStreamLoader
1066: Function GetDesktopDirectoryFolder : string
1067: Function GetDesktopFolder : string
1068: Function GetDFAState( oStates : TList) : TniRegularExpressionState
1069: Function GetDirectorySize( const Path : string) : Int64
1070: Function GetDisplayWidth : Integer
1071: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer) : Boolean
1072: Function GetDomainName : string
1073: Function GetDriverRegistryFile( DesignMode : Boolean) : string
1074: function GetDriveType(rootpath: pchar): cardinal;
1075: Function GetDriveTypeStr( const Drive : Char) : string
1076: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1077: Function GetEnumerator : TListItemsEnumerator
1078: Function GetEnumerator : TTaskDialogButtonsEnumerator
1079: Function GetEnumerator : TToolBarEnumerator
1080: Function GetEnumerator : TTreeNodesEnumerator
1081: Function GetEnumerator : TWideStringsEnumerator
1082: Function GetEnvVar( const VarName : string) : string
1083: Function GetEnvironmentVar( const AVariableName : string) : string
1084: Function GetEnvironmentVariable( const VarName : string) : string
1085: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean) : Boolean
1086: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean) : Boolean
1087: Function getEnvironmentString: string;
1088: Function GetExceptionHandler : TObject
1089: Function GetFavoritesFolder : string
1090: Function GetFieldByName( const Name : string) : string
1091: Function GetFieldInfo( const Origin : Widestring; var FieldInfo : TFieldInfo) : Boolean
1092: Function GetFieldValue( ACol : Integer) : string
1093: Function GetFileAgeCoherence( const FileName : string) : Boolean
1094: Function GetFileCreation( const FileName : string) : TFileTime
1095: Function GetFileCreationTime( const Filename : string) : TDateTime
1096: Function GetFileInformation( const FileName : string) : TSearchRec
1097: Function GetFileLastAccess( const FileName : string) : TFileTime
1098: Function GetFileLastWrite( const FileName : string) : TFileTime
1099: Function GetFileList(FileList: TStringlist; apath: string): TStringlist;
1100: Function GetFileList1(apath: string): TStringlist;
1101: Function GetFileMIMEType( const AFileName : string) : string
1102: Function GetFileSize( const FileName : string) : Int64
1103: Function GetFileVersion( AFileName : string) : Cardinal
1104: Function GetFileVersion( const AFilename : string) : Cardinal
1105: Function GetFileSize2(Handle: Integer; x: Integer): Integer; stdcall;
1106: Function GetFileDate(aFile:string; aWithTime:Boolean):string;
1107: Function GetFilterData( Root : PExprNode) : TExprData
1108: Function getFirstChild : LongInt
1109: Function getFirstChild : TTreeNode
1110: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string) : string
1111: Function GetFirstNode : TTreeNode
1112: Function GetFontsFolder : string
1113: Function GetFormulaValue( const Formula : string) : Extended
1114: Function GetFreePageFileMemory : Integer
1115: Function GetFreePhysicalMemory : Integer
1116: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind) : Integer;
1117: Function GetFreeSystemResources1 : TFreeSystemResources;
1118: Function GetFreeVirtualMemory : Integer
1119: Function GetFromClipboard : Boolean
1120: Function GetFullURI( const AOptionalFileds : TIdURIOptionalFieldsSet) : String
1121: Function GetGBitmap( Value : TBitmap) : TBitmap
1122: Function GetGMTDateByName( const AFileName : TIdFileName) : TDateTime
1123: Function GetGroupState( Level : Integer) : TGroupPosInds
1124: Function GetHandle : HWND
1125: Function GETHELPCONTEXT( VALUE : INTEGER; BYCOMMAND : BOOLEAN) : THELPCONTEXT
1126: function GetHexArray(ahexdig: THexArray): THexArray;
1127: Function GetHighLightColor( const Color : TColor; Luminance : Integer) : TColor
1128: function GetHINSTANCE: longword;
1129: Function GetHistoryFolder : string
1130: Function GetHitTestInfoAt( X, Y : Integer) : THitTests
1131: function getHMODULE: longword;
1132: Function GetHostByName(const AComputerName: String): String;
1133: Function GetHostName : string
1134: Function getHostIP: string;
1135: Function GetHotSpot : TPoint
1136: Function GetHueBitmap( Value : TBitmap) : TBitmap
1137: Function GetImageBitmap : HBITMAP
1138: Function GETIMAGELIST : TCUSTOMIMAGELIST
1139: Function GetIncome( const aNetto : Currency) : Currency
```

```
1140: Function GetIncome( const aNetto : Extended) : Extended
1141: Function GetIncome( const aNetto : Extended): Extended
1142: Function GetIncome(const aNetto : Extended) : Extended
1143: function GetIncome(const aNetto: Currency): Currency
1144: Function GetIncome2( const aNetto : Currency) : Currency
1145: Function GetIncome2( const aNetto : Currency): Currency
1146: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string) : string
1147: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN) : TINDEXDEF
1148: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet) : TIndexDef
1149: Function GetInstRes(Instance:THandle;ResType:TResType;const
      Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1150: Function
      GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Integer;LoadFlags:TLoadResources;MaskColor:
      TColor):Boolean;
1151: Function GetIntelCacheDescription( const D : Byte) : string
1152: Function GetInteractiveUserName : string
1153: Function GetInternetCacheFolder : string
1154: Function GetInternetFormattedFileTimeStamp( const AFilename : String) : String
1155: Function GetIPAddress( const HostName : string) : string
1156: Function GetIP( const HostName : string) : string
1157: Function GetIPHostByName(const AComputerName: String): String;
1158: Function GetIsAdmin: Boolean;
1159: Function GetItem( X, Y : Integer) : LongInt
1160: Function GetItemAt( X, Y : Integer) : TListItem
1161: Function GetItemHeight(Font: TFont): Integer;
1162: Function GetItemPath( Index : Integer) : string
1163: Function GetKeyFieldNames( List : TStrings) : Integer;
1164: Function GetKeyFieldNames1( List : TWideStrings) : Integer;
1165: Function GetKeyState( const VirtualKey : Cardinal) : Boolean
1166: Function GetLastChild : LongInt
1167: Function GetLastChild : TTreeNode
1168: Function GetLastDelimitedToken( const cDelim : char; const cStr : string) : string
1169: function GetLastError: Integer
1170: Function GetLAT_CONV_FACTOR: double;  //for WGS84 power(1 - 1 / 298.257223563, 2);
1171: Function GetLoader( Ext : string) : TBitmapLoader
1172: Function GetLoadFilter : string
1173: Function GetLocalComputerName : string
1174: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char) : Char
1175: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string) : string
1176: Function GetLocalUserName : string
1177: Function GetLoginUsername : WideString
1178: function getLongDayNames: string)
1179: Function GetLongHint(const hint: string): string
1180: function getLongMonthNames: string)
1181: Function GetMacAddresses( const Machine : string; const Addresses : TStrings) : Integer
1182: Function GetMainAppWndFromPid( PID : DWORD) : HWND
1183: Function GetMaskBitmap : HBITMAP
1184: Function GetMaxAppAddress : Integer
1185: Function GetMciErrorMessage( const MciErrNo : MCIERROR) : string
1186: Function GetMemoryLoad : Byte
1187: Function GetMIMEDefaultFileExt( const MIMEType : string) : TIdFileName
1188: Function GetMIMETypeFromFile( const AFile : string) : string
1189: Function GetMIMETypeFromFile( const AFile : TIdFileName) : string
1190: Function GetMinAppAddress : Integer
1191: Function GetModule : TComponent
1192: Function GetModuleHandle( ModuleName : PChar) : HMODULE
1193: Function GetModuleName( Module : HMODULE) : string
1194: Function GetModulePath( const Module : HMODULE) : string
1195: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;
1196: Function GetCommandLine: PChar; stdcall;
1197: Function GetMonochromeBitmap( Value : TBitmap) : TBitmap
1198: Function GetMultiN(aval: integer): string;
1199: Function GetName : String
1200: Function GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1201: Function GetNethoodFolder : string
1202: Function GetNext : TTreeNode
1203: Function GetNextChild( Value : LongInt) : LongInt
1204: Function GetNextChild( Value : TTreeNode) : TTreeNode
1205: Function GetNextDelimitedToken( const cDelim : char; var cStr : String) : String
1206: Function GetNextItem( StartItem: TListItem;Direction:TSearchDirection;States:TItemStates) : TListItem
1207: Function GetNextPacket : Integer
1208: Function getNextSibling : TTreeNode
1209: Function GetNextVisible : TTreeNode
1210: Function GetNode( ItemId : HTreeItem) : TTreeNode
1211: Function GetNodeAt( X, Y : Integer) : TTreeNode
1212: Function GetNodeDisplayWidth( Node : TOutlineNode) : Integer
1213: function GetNumberOfProcessors: longint;
1214: Function GetNumLockKeyState : Boolean
1215: Function GetObjectProperty( Instance : TPersistent; const PropName : string) : TObject
1216: Function GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1217: Function GetOptionalParam( const ParamName : string) : OleVariant
1218: Function GetOSName: string;
1219: Function GetOSVersion: string;
1220: Function GetOSNumber: string;
1221: Function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
1222: Function GetPackageModuleHandle( PackageName : PChar) : HMODULE
1223: function GetPageSize: Cardinal;
1224: Function GetParameterFileName : string
1225: Function GetParams( var OwnerData : OleVariant) : OleVariant
```

```
1226: Function GETPARENTCOMPONENT : TCOMPONENT
1227: Function GetParentForm(control: TControl): TForm
1228: Function GETPARENTMENU : TMENU
1229: Function GetPassword : Boolean
1230: Function GetPassword : string
1231: Function GetPersonalFolder : string
1232: Function GetPidFromProcessName( const ProcessName : string) : DWORD
1233: Function GetPosition : TPoint
1234: Function GetPrev : TTreeNode
1235: Function GetPrevChild( Value : LongInt) : LongInt
1236: Function GetPrevChild( Value : TTreeNode) : TTreeNode
1237: Function getPrevSibling : TTreeNode
1238: Function GetPrevVisible : TTreeNode
1239: Function GetPrinthoodFolder : string
1240: Function GetPrivilegeDisplayName( const PrivilegeName : string) : string
1241: Function getProcessList: TStrings;
1242: Function GetProcessId : TIdPID
1243: Function GetProcessNameFromPid( PID : DWORD) : string
1244: Function GetProcessNameFromWnd( Wnd : HWND) : string
1245: Function GetProcessMemoryInfo(Process: THandle;ppsmemCounters: TProcessMemoryCounters;cb: DWORD):BOOL
1246: Function getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1247: Function getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1248: Function GetProgramFilesFolder : string
1249: Function GetProgramsFolder : string
1250: Function GetProxy : string
1251: Function GetQuoteChar : WideString
1252: Function GetQrCode4(Width,Height:Word; Correct_Level:string;const
      Data:string;aformat:string):TLinearBitmap;
1253: Function GetQrCodetoFile(Width,Height:Word;Correct_Level:string;const
      Data:string;aformat:string):TLinearBitmap;
1254: Function GetRate : Double
1255: Function getPerfTime: string;
1256: Function getRuntime: string;
1257: Function GetRBitmap( Value : TBitmap) : TBitmap
1258: Function GetReadableName( const AName : string) : string
1259: Function GetRecentDocs : TStringList
1260: Function GetRecentFolder : string
1261: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer) : OleVariant;
1262: Function GetRecords1(Count:Integer; out RecsOut:Integer;Options:Integer;const CommandText:WideString;var
      Params, OwnerData : OleVariant) : OleVariant;
1263: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString) : _Recordset
1264: Function GetRegisteredCompany : string
1265: Function GetRegisteredOwner : string
1266: Function GetResource(ResType:TResType;const
      Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor:Boolean
1267: Function GetResourceName( ObjStream : TStream; var AName : string) : Boolean
1268: Function GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1269: Function GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1270: Function GetRValue( rgb : DWORD) : Byte
1271: Function GetGValue( rgb : DWORD) : Byte
1272: Function GetBValue( rgb : DWORD) : Byte
1273: Function GetCValue( cmyk : COLORREF) : Byte
1274: Function GetMValue( cmyk : COLORREF) : Byte
1275: Function GetYValue( cmyk : COLORREF) : Byte
1276: Function GetKValue( cmyk : COLORREF) : Byte
1277: Function CMYK( c, m, y, k : Byte) : COLORREF
1278: Function GetOSName: string;
1279: Function GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR) : FARPROC
1280: Function GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1281: Function GetSafeCallExceptionMsg : String
1282: Function GetSaturationBitmap( Value : TBitmap) : TBitmap
1283: Function GetSaveFilter : string
1284: Function GetSaver( Ext : string) : TBitmapLoader
1285: Function GetScrollLockKeyState : Boolean
1286: Function GetSearchString : string
1287: Function GetSelections( AList : TList) : TTreeNode
1288: function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1289: Function GetSendToFolder : string
1290: Function GetServer : IAppServer
1291: Function GetServerList : OleVariant
1292: Function GetShadowColor( const Color : TColor; Luminance : Integer) : TColor
1293: Function GetShellProcessHandle : THandle
1294: Function GetShellProcessName : string
1295: Function GetShellVersion : Cardinal
1296: function getShortDayNames: string)
1297: Function GetShortHint(const hint: string): string
1298: function getShortMonthNames: string)
1299: Function GetSizeOfFile( const FileName : string) : Int64;
1300: Function GetSizeOfFile1( Handle : THandle) : Int64;
1301: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1302: Function GetStartmenuFolder : string
1303: Function GetStartupFolder : string
1304: Function GetStringProperty( Instance : TPersistent; const PropName : string) : WideString
1305: Function GetSuccessor( cChar : char) : TniRegularExpressionState
1306: Function GetSwapFileSize : Integer
1307: Function GetSwapFileUsage : Integer
1308: Function GetSystemLocale : TIdCharSet
1309: Function GetSystemMetrics( nIndex : Integer) : Integer
1310: Function GetSystemPathSH(Folder: Integer): TFilename ;
```

```
1311: Function GetTableNameFromQuery( const SQL : Widestring) : Widestring
1312: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1313: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1314: Function GetTasksList( const List : TStrings) : Boolean
1315: Function getTeamViewerID: string;
1316: Function GetTemplatesFolder : string
1317: Function GetText : PwideChar
1318: function GetText:PChar
1319: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1320: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1321: Function GetTextItem( const Value : string) : Longint
1322: function GETTEXTLEN:INTEGER
1323: Function GetThreadLocale: Longint; stdcall
1324: Function GetCurrentThreadID: LongWord; stdcall;
1325: Function GetTickCount : Cardinal
1326: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal) : Cardinal
1327: Function GetTicketNr : longint
1328: Function GetTime : Cardinal
1329: Function GetTime : TDateTime
1330: Function GetTimeout : Integer
1331: Function GetTimeStr: String
1332: Function GetTimeString: String
1333: Function GetTodayFiles(startdir, amask: string): TStringlist;
1334: Function getTokenCounts : integer
1335: Function GetTotalPageFileMemory : Integer
1336: Function GetTotalPhysicalMemory : Integer
1337: Function GetTotalVirtualMemory : Integer
1338: Function GetUniqueFileName( const APath, APrefix, AExt : String) : String
1339: Function GetUseNowForDate : Boolean
1340: Function GetUserDomainName( const CurUser : string) : string
1341: Function GetUserName : string
1342: Function GetUserName: string;
1343: Function GetUserObjectName( hUserObject : THandle) : string
1344: Function GetValueBitmap( Value : TBitmap) : TBitmap
1345: Function GetValueMSec : Cardinal
1346: Function GetValueStr : String
1347: Function GetVersion: int;
1348: Function GetVersionString(FileName: string): string;
1349: Function GetVisibleNode( Index : LongInt) : TOutlineNode
1350: Function GetVolumeFileSystem( const Drive : string) : string
1351: Function GetVolumeName( const Drive : string) : string
1352: Function GetVolumeSerialNumber( const Drive : string) : string
1353: Function GetWebAppServices : IWebAppServices
1354: Function GetWebRequestHandler : IWebRequestHandler
1355: Function GetWindowCaption( Wnd : HWND) : string
1356: Function GetWindowDC(hdwnd: HWND): HDC;
1357: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1358: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1359: Function GetWindowsComputerID : string
1360: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1361: Function GetWindowsFolder : string
1362: Function GetWindowsServicePackVersion : Integer
1363: Function GetWindowsServicePackVersionString : string
1364: Function GetWindowsSystemFolder : string
1365: Function GetWindowsTempFolder : string
1366: Function GetWindowsUserID : string
1367: Function GetWindowsVersion : TWindowsVersion
1368: Function GetWindowsVersionString : string
1369: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1370: Function GMTToLocalDateTime( S : string) : TDateTime
1371: Function GotoKey : Boolean
1372: Function GradToCycle( const Grads : Extended) : Extended
1373: Function GradToDeg( const Grads : Extended) : Extended
1374: Function GradToDeg( const Value : Extended) : Extended;
1375: Function GradToDeg1( const Value : Double) : Double;
1376: Function GradToDeg2( const Value : Single) : Single;
1377: Function GradToRad( const Grads : Extended) : Extended
1378: Function GradToRad( const Value : Extended) : Extended;
1379: Function GradToRad1( const Value : Double) : Double;
1380: Function GradToRad2( const Value : Single) : Single;
1381: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1382: Function GreenComponent( const Color32 : TColor32) : Integer
1383: function GUIDToString(const GUID: TGUID): string)
1384: Function HandleAllocated : Boolean
1385: function HandleAllocated: Boolean;
1386: Function HandleRequest : Boolean
1387: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1388: Function HarmonicMean( const X : TDynFloatArray) : Float
1389: Function HasAsParent( Value : TTreeNode) : Boolean
1390: Function HASCHILDDEFS : BOOLEAN
1391: Function HasCurValues : Boolean
1392: Function HasExtendCharacter( const s : UTF8String) : Boolean
1393: Function HasFormat( Format : Word) : Boolean
1394: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1395: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1396: Function HashValue(AStream: TStream): LongWord
1397: Function HashValue(AStream: TStream): Word
1398: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1399: Function HashValue1(AStream : TStream): T4x4LongWordRecord
```

```
1400: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1401: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1402: Function HashValue16( const ASrc : string) : Word;
1403: Function HashValue16stream( AStream : TStream) : Word;
1404: Function HashValue32( const ASrc : string) : LongWord;
1405: Function HashValue32Stream( AStream : TStream) : LongWord;
1406: Function HasMergeConflicts : Boolean
1407: Function hasMoreTokens : boolean
1408: Function HASPARENT : BOOLEAN
1409: function HasParent: Boolean
1410: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1411: Function HasUTF8BOM( S : TStream) : boolean;
1412: Function HasUTF8BOM1( S : AnsiString) : boolean;
1413: Function Haversine( X : Float) : Float
1414: Function Head( s : string; const subs : string; var tail : string) : string
1415: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1416: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1417: function HELPJUMP(JUMPID:STRING):BOOLEAN
1418: Function HeronianMean( const a, b : Float) : Float
1419: function HexStrToStr(Value: string): string;
1420: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1421: function HexToBin2(HexNum: string): string;
1422: Function HexToDouble( const Hex : string) : Double
1423: function HexToInt(hexnum: string): LongInt;
1424: function HexToStr(Value: string): string;
1425: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1426: function Hi(vdat: word): byte;
1427: function HiByte(W: Word): Byte)
1428: function High: Int64;
1429: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1430: function HINSTANCE: longword;
1431: function HiWord(l: DWORD): Word)
1432: function HMODULE: longword;
1433: Function HourOf( const AValue : TDateTime) : Word
1434: Function HourOfTheDay( const AValue : TDateTime) : Word
1435: Function HourOfTheMonth( const AValue : TDateTime) : Word
1436: Function HourOfTheWeek( const AValue : TDateTime) : Word
1437: Function HourOfTheYear( const AValue : TDateTime) : Word
1438: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1439: Function HourSpan( const ANow, AThen : TDateTime) : Double
1440: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1441: Function HTMLDecode( const AStr : String) : String
1442: Function HTMLEncode( const AStr : String) : String
1443: Function HTMLEscape( const Str : string) : string
1444: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1445: Function HTTPDecode( const AStr : String) : string
1446: Function HTTPEncode( const AStr : String) : string
1447: Function Hypot( const X, Y : Extended) : Extended
1448: Function IBMax( n1, n2 : Integer) : Integer
1449: Function IBMin( n1, n2 : Integer) : Integer
1450: Function IBRandomString( iLength : Integer) : String
1451: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1452: Function IBStripString( st : String; CharsToStrip : String) : String
1453: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1454: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1455: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1456: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1457: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1458: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1459:  Function RandomString( iLength : Integer) : String');
1460:  Function RandomInteger( iLow, iHigh : Integer) : Integer');
1461:  Function StripString( st : String; CharsToStrip : String) : String');
1462:  Function FormatIdentifier( Dialect : Integer; Value : String) : String');
1463:  Function FormatIdentifierValue( Dialect : Integer; Value : String) : String');
1464:  Function ExtractIdentifier( Dialect : Integer; Value : String) : String');
1465:  Function QuoteIdentifier( Dialect : Integer; Value : String) : String');
1466:  Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string;
1467:  Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String;
1468:  function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1469: Function IconToBitmap( Ico : HICON) : TBitmap
1470: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1471: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1472: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean)
1473: function IdentToColor(const Ident: string; var Color: Longint): Boolean)
1474: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1475: Function IdGetDefaultCharSet : TIdCharSet
1476: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1477: Function IdPorts2 : TStringList
1478: Function IdToMib( const Id : string) : string
1479: Function IdSHA1Hash(apath: string): string;
1480: Function IdHashSHA1(apath: string): string;
1481: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1482: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1483: Function IfThenInt( AValue : Boolean; const ATrue : integer; AFalse : integer): integer;');
1484: Function IfThenDouble( AValue : Boolean; const ATrue : double; AFalse : double): double;');
1485: Function IfThenBool( AValue : Boolean; const ATrue : boolean; AFalse : boolean): boolean;');
1486: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
1487: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1488: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean) : Boolean;
```

```
1489: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1490: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer) : TDateTime
1491: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1492: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1493: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1494: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1495: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1496: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1497: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1498: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1499: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1500: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1501: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1502: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1503: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1504: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1505: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1506: Function IncludeTrailingBackslash( S : string) : string
1507: function IncludeTrailingBackslash(const S: string): string)
1508: Function IncludeTrailingPathDelimiter( const APath : string) : string
1509: Function IncludeTrailingPathDelimiter( S : string) : string
1510: function IncludeTrailingPathDelimiter(const S: string): string)
1511: Function IncludeTrailingSlash( const APath : string) : string
1512: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliSeconds : Int64) : TDateTime
1513: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64) : TDateTime
1514: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer) : TDateTime
1515: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1516: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64) : TDateTime
1517: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer) : TDateTime
1518: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer) : TDateTime
1519: Function IndexOf( AClass : TClass) : Integer
1520: Function IndexOf( AComponent : TComponent) : Integer
1521: Function IndexOf( AObject : TObject) : Integer
1522: Function INDEXOF( const ANAME : String) : INTEGER
1523: Function IndexOf( const DisplayName : string) : Integer
1524: Function IndexOf( const Item : TBookmarkStr) : Integer
1525: Function IndexOf( const S : WideString) : Integer
1526: Function IndexOf( const View : TJclFileMappingView) : Integer
1527: Function INDEXOF( FIELD : TFIELD) : INTEGER
1528: Function IndexOf( ID : LCID) : Integer
1529: Function INDEXOF( ITEM : TMENUITEM) : INTEGER
1530: Function IndexOf( Value : TListItem) : Integer
1531: Function IndexOf( Value : TTreeNode) : Integer
1532: function IndexOf(const S: string): Integer;
1533: Function IndexOfName( const Name : WideString) : Integer
1534: function IndexOfName(Name: string): Integer;
1535: Function IndexOfObject( AObject : TObject) : Integer
1536: function IndexofObject(AObject:tObject):Integer
1537: Function IndexOfTabAt( X, Y : Integer) : Integer
1538: Function IndexStr( const AText : string; const AValues : array of string) : Integer
1539: Function IndexText( const AText : string; const AValues : array of string) : Integer
1540: Function IndexOfInteger( AList : TStringList; Value : Variant) : Integer
1541: Function IndexOfFloat( AList : TStringList; Value : Variant) : Integer
1542: Function IndexOfDate( AList : TStringList; Value : Variant) : Integer
1543: Function IndexOfString( AList : TStringList; Value : Variant) : Integer
1544: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1545: Function IndyGetHostName : string
1546: Function IndyInterlockedDecrement( var I : Integer) : Integer
1547: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1548: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
1549: Function IndyInterlockedIncrement( var I : Integer) : Integer
1550: Function IndyLowerCase( const A1 : string) : string
1551: Function IndyStrToBool( const AString : String) : Boolean
1552: Function IndyUpperCase( const A1 : string) : string
1553: Function InitCommonControl( CC : Integer) : Boolean
1554: Function InitTempPath : string
1555: Function InMainThread : boolean
1556: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1557: Function Input: Text)
1558: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1559: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1560: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1561: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1562: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1563: Function InquireSignal( RtlSigNum : Integer) : TSignalState
1564: Function InRangeR( const A, Min, Max : Double) : Boolean
1565: function Insert( Index : Integer) : TCollectionItem
1566: Function Insert( Index : Integer) : TComboExItem
1567: Function Insert( Index : Integer) : THeaderSection
1568: Function Insert( Index : Integer) : TListItem
1569: Function Insert( Index : Integer) : TStatusPanel
1570: Function Insert( Index : Integer) : TWorkArea
1571: Function Insert( Index : LongInt; const Text : string) : LongInt
1572: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1573: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1574: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1575: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1576: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
1577: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
```

```
1578: Function Instance : Longint
1579: function InstanceSize: Longint
1580: Function Int(e : Extended) : Extended;
1581: function Int64ToStr(i: Int64): String;
1582: Function IntegerToBcd( const AValue : Integer) : TBcd
1583: Function Intensity( const Color32 : TColor32) : Integer;
1584: Function Intensity( const R, G, B : Single) : Single;
1585: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
      FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1586: Function InterestRate(NPeriods:Integer;const Payment,PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime): Extended
1587: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1588: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1589: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1590: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1591: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1592: Function IntMibToStr( const Value : string) : string
1593: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1594: Function IntToBin( Value : cardinal) : string
1595: Function IntToHex( Value : Integer; Digits : Integer) : string;
1596: function IntToHex(a: integer; b: integer): string;
1597: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1598: function IntToHex64(Value: Int64; Digits: Integer): string)
1599: Function IntTo3Str( Value : Longint; separator: string) : string
1600: Function inttobool( aInt : LongInt) : Boolean
1601: function IntToStr(i: Int64): String;
1602: Function IntToStr64(Value: Int64): string)
1603: function IOResult: Integer
1604: Function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1605: Function IsAccel(VK: Word; const Str: string): Boolean
1606: Function IsAddressInNetwork( Address : String) : Boolean
1607: Function IsAdministrator : Boolean
1608: Function IsAlias( const Name : string) : Boolean
1609: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1610: Function IsASCII( const AByte : Byte) : Boolean;
1611: Function IsASCIILDH( const AByte : Byte) : Boolean;
1612: Function IsAssembly(const FileName: string): Boolean;
1613: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1614: Function IsBinary(const AChar : Char) : Boolean
1615: function IsConsole: Boolean)
1616: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1617: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1618: Function IsDelphiDesignMode : boolean
1619: Function IsDelphiRunning : boolean
1620: Function IsDFAState : boolean
1621: Function IsDirectory( const FileName : string) : Boolean
1622: Function IsDomain( const S : String) : Boolean
1623: function IsDragObject(Sender: TObject): Boolean;
1624: Function IsEditing : Boolean
1625: Function ISEMPTY : BOOLEAN
1626: Function IsEqual( Value : TParameters) : Boolean
1627: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1628: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1629: Function IsFirstNode : Boolean
1630: Function IsFloatZero( const X : Float) : Boolean
1631: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1632: Function IsFormOpen(const FormName: string): Boolean;
1633: Function IsFQDN( const S : String) : Boolean
1634: Function IsGrayScale : Boolean
1635: Function IsHex( AChar : Char) : Boolean;
1636: Function IsHexString(const AString: string): Boolean;
1637: Function IsHostname( const S : String) : Boolean
1638: Function IsInfinite( const AValue : Double) : Boolean
1639: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1640: Function IsInternet: boolean;
1641: Function IsLeadChar( ACh : Char) : Boolean
1642: Function IsLeapYear( Year : Word) : Boolean
1643: function IsLeapYear(Year: Word): Boolean)
1644: function IsLibrary: Boolean)
1645: Function ISLINE : BOOLEAN
1646: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1647: Function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1648: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1649: Function IsMatch( const Pattern, Text : string) : Boolean  //Grep like RegEx
1650: Function IsMainAppWindow( Wnd : HWND) : Boolean
1651: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1652: function IsMemoryManagerSet: Boolean)
1653: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1654: function IsMultiThread: Boolean)
1655: Function IsNumeric( AChar : Char) : Boolean;
1656: Function IsNumeric2( const AString : string) : Boolean;
1657: Function IsOctal( AChar : Char) : Boolean;
1658: Function IsOctalString(const AString: string) : Boolean;
1659: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1660: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1661: Function IsPM( const AValue : TDateTime) : Boolean
1662: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1663: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1664: Function IsPrimeRM( N : Cardinal) : Boolean  //rabin miller
```

```
1665: Function IsPrimeTD( N : Cardinal) : Boolean  //trial division
1666: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1667: Function ISqrt( const I : Smallint) : Smallint
1668: Function IsReadOnly(const Filename: string): boolean;
1669: Function IsRectEmpty( const Rect : TRect) : Boolean
1670: function IsRectEmpty(const Rect: TRect): Boolean)
1671: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1672: Function ISRIGHTTOLEFT : BOOLEAN
1673: function IsRightToLeft: Boolean
1674: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1675: Function ISSEQUENCED : BOOLEAN
1676: Function IsSystemModule( const Module : HMODULE) : Boolean
1677: Function IsSystemResourcesMeterPresent : Boolean
1678: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1679: Function IsToday( const AValue : TDateTime) : Boolean
1680: function IsToday(const AValue: TDateTime): Boolean;
1681: Function IsTopDomain( const AStr : string) : Boolean
1682: Function IsUTF8LeadByte( Lead : Char) : Boolean
1683: Function IsUTF8String( const s : UTF8String) : Boolean
1684: Function IsUTF8TrailByte( Lead : Char) : Boolean
1685: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1686: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1687: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1688: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1689: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1690: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1691: Function IsValidIdent( Ident : string) : Boolean
1692: Function IsValidIdent(const Ident: string; AllowDots: Boolean): Boolean)
1693: Function IsValidIP( const S : String) : Boolean
1694: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean
1695: Function IsValidISBN( const ISBN : AnsiString) : Boolean
1696: Function IsVariantManagerSet: Boolean; //deprecated;
1697: Function IsVirtualPcGuest : Boolean;
1698: Function IsVmWareGuest : Boolean;
1699: Function IsVCLControl(Handle: HWnd): Boolean;
1700: Function IsWhiteString( const AStr : String) : Boolean
1701: Function IsWindowResponding( Wnd : HWND; Timeout : Integer) : Boolean
1702: Function IsWoW64: boolean;
1703: Function IsWin64: boolean;
1704: Function IsWow64String(var s: string): Boolean;
1705: Function IsWin64String(var s: string): Boolean;
1706: Function IsWindowsVista: boolean;
1707: Function isPowerof2(num: int64): boolean;
1708: Function powerOf2(exponent: integer): int64;
1709: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1710: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1711: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1712: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1713: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1714: Function ItemRect( Index : Integer) : TRect
1715: function ITEMRECT(INDEX:INTEGER):TRECT
1716: Function ItemWidth( Index : Integer) : Integer
1717: Function JavahashCode(val: string): Integer;
1718: Function JosephusG(n,k: integer; var graphout: string): integer;
1719: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1720: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1721: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1722: Function KeepAlive : Boolean
1723: Function KeysToShiftState(Keys: Word): TShiftState;
1724: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1725: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1726: Function KeyboardStateToShiftState: TShiftState; overload;
1727: Function Languages : TLanguages
1728: Function Last : TClass
1729: Function Last : TComponent
1730: Function Last : TObject
1731: Function LastDelimiter( Delimiters, S : string) : Integer
1732: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1733: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1734: Function Latitude2WGS84(lat: double): double;
1735: Function LCM(m,n:longint):longint;
1736: Function LCMJ( const X, Y : Cardinal) : Cardinal
1737: Function Ldexp( const X : Extended; const P : Integer) : Extended
1738: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1739: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1740: function Length: Integer;
1741: Procedure LetFileList(FileList: TStringlist; apath: string);
1742: function lengthmp3(mp3path: string):integer;
1743: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1744: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1745: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1746: function LineStart(Buffer, BufPos: PChar): PChar
1747: function LineStart(Buffer, BufPos: PChar): PChar)
1748: function ListSeparator: char;
1749: function Ln(x: Extended): Extended;
1750: Function LnXP1( const X : Extended) : Extended
1751: function Lo(vdat: word): byte;
1752: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
```

```
1753: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1754: Function LoadFileAsString( const FileName : string) : string
1755: Function LoadFromFile( const FileName : string) : TBitmapLoader
1756: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1757: Function LoadPackage(const Name: string): HMODULE
1758: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1759: Function LoadStr( Ident : Integer) : string
1760: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1761: Function LoadWideStr( Ident : Integer) : WideString
1762: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1763: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1764: Function LockServer( fLock : LongBool) : HResult
1765: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1766: Function Log( const X : Extended) : Extended
1767: Function Log10( const X : Extended) : Extended
1768: Function Log2( const X : Extended) : Extended
1769: function LogBase10(X: Float): Float;
1770: Function LogBase2(X: Float): Float;
1771: Function LogBaseN(Base, X: Float): Float;
1772: Function LogN( const Base, X : Extended) : Extended
1773: Function LogOffOS : Boolean
1774: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1775: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1776: Function LongDateFormat: string;
1777: function LongTimeFormat: string;
1778: Function LongWordToFourChar( ACardinal : LongWord) : string
1779: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1780: Function LookupName( const name : string) : TInAddr
1781: Function LookupService( const service : string) : Integer
1782: function Low: Int64;
1783: Function LowerCase( S : string) : string
1784: Function Lowercase(s : AnyString) : AnyString;
1785: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1786: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1787: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1788: function MainInstance: longword
1789: function MainThreadID: longword
1790: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1791: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1792: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1793: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1794: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1795: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1796: function MakeLong(A, B: Word): Longint)
1797: Function MakeTempFilename( const APath : String) : string
1798: Function MakeValidFileName( const Str : string) : string
1799: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1800: function MakeWord(A, B: Byte): Word)
1801: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1802: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1803: Function MapValues( Mapping : string; Value : string) : string
1804: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1805: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1806: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1807: Function MaskGetFldSeparator( const EditMask : string) : Integer
1808: Function MaskGetMaskBlank( const EditMask : string) : Char
1809: Function MaskGetMaskSave( const EditMask : string) : Boolean
1810: Function MaskIntlLiteralToChar( IChar : Char) : Char
1811: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1812: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1813: Function MaskString( Mask, Value : String) : String
1814: Function Match( const sString : string) : TniRegularExpressionMatchResul
1815: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1816: Function Matches( const Filename : string) : Boolean
1817: Function MatchesMask( const Filename, Mask : string) : Boolean
1818: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1819: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1820: Function Max( AValueOne, AValueTwo : Integer) : Integer
1821: function Max(const x,y: Integer): Integer;
1822: Function Max1( const B1, B2 : Shortint) : Shortint;
1823: Function Max2( const B1, B2 : Smallint) : Smallint;
1824: Function Max3( const B1, B2 : Word) : Word;
1825: function Max3(const x,y,z: Integer): Integer;
1826: Function Max4( const B1, B2 : Integer) : Integer;
1827: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1828: Function Max6( const B1, B2 : Int64) : Int64;
1829: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1830: Function MaxFloat( const X, Y : Float) : Float
1831: Function MaxFloatArray( const B : TDynFloatArray) : Float
1832: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1833: function MaxIntValue(const Data: array of Integer):Integer)
1834: Function MaxJ( const B1, B2 : Byte) : Byte;
1835: function MaxPath: string;
1836: function MaxValue(const Data: array of Double): Double)
1837: Function MaxCalc( const Formula : string) : Extended  //math expression parser
1838: Procedure MaxCalcF( const Formula : string);   //out to console memo2
1839: function MD5(const fileName: string): string;
1840: Function Mean( const Data : array of Double) : Extended
1841: Function Median( const X : TDynFloatArray) : Float
```

```
1842: Function Memory : Pointer
1843: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1844: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1845: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1846: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1847: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1848: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1849: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1850: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1851: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1852: Function MibToId( Mib : string) : string
1853: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1854: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1855: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1856: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1857: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1858: Procedure GetMidiOutputs( const List : TStrings)
1859: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1860: Function MIDINoteToStr( Note : TMIDINote) : string
1861: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1862: Procedure GetMidiOutputs( const List : TStrings)
1863: Procedure MidiOutCheck( Code : MMResult)
1864: Procedure MidiInCheck( Code : MMResult)
1865: Function MilliSecondOf( const AValue : TDateTime) : Word
1866: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1867: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1868: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1869: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1870: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1871: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
1872: Function MilliSecondOfTheYear( const AValue : TDateTime) : Int64
1873: Function MilliSecondsBetween( const ANow, AThen : TDateTime) : Int64
1874: Function MilliSecondSpan( const ANow, AThen : TDateTime) : Double
1875: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
1876: Function millis: int64;
1877: Function Min( AValueOne, AValueTwo : Integer) : Integer
1878: Function Min1( const B1, B2 : Shortint) : Shortint;
1879: Function Min2( const B1, B2 : Smallint) : Smallint;
1880: Function Min3( const B1, B2 : Word) : Word;
1881: Function Min4( const B1, B2 : Integer) : Integer;
1882: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1883: Function Min6( const B1, B2 : Int64) : Int64;
1884: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1885: Function MinClientRect : TRect;
1886: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1887: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1888: Function MinFloat( const X, Y : Float) : Float
1889: Function MinFloatArray( const B : TDynFloatArray) : Float
1890: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1891: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1892: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1893: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1894: Function MinIntValue( const Data : array of Integer) : Integer
1895: function MinIntValue(const Data: array of Integer):Integer)
1896: Function MinJ( const B1, B2 : Byte) : Byte;
1897: Function MinuteOf( const AValue : TDateTime) : Word
1898: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1899: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1900: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1901: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1902: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1903: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1904: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1905: Function MinValue( const Data : array of Double) : Double
1906: function MinValue(const Data: array of Double): Double)
1907: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1908: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1909: Function ModFloat( const X, Y : Float) : Float
1910: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1911: Function Modify( const Key : string; Value : Integer) : Boolean
1912: Function ModuleCacheID : Cardinal
1913: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1914: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1915: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1916: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1917: Function MonthOf( const AValue : TDateTime) : Word
1918: Function MonthOfTheYear( const AValue : TDateTime) : Word
1919: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1920: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1921: Function MonthStr( DateTime : TDateTime) : string
1922: Function MouseCoord( X, Y : Integer) : TGridCoord
1923: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1924: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1925: Function MoveNext : Boolean
```

```
1926: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1927: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1928: Function Name : string
1929: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1930: function NetworkVolume(DriveChar: Char): string
1931: Function NEWBOTTOMLINE : INTEGER
1932: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1933: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1934: Function NEWLINE : TMENUITEM
1935: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1936: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1937: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
      const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1938: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1939: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
      TMenuItem;AENABLED:BOOL): TMENUITEM
1940: Function NEWTOPLINE : INTEGER
1941: Function Next : TIdAuthWhatsNext
1942: Function NextCharIndex( S : String; Index : Integer) : Integer
1943: Function NextRecordSet : TCustomSQLDataSet
1944: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1945: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1946: Function NextToken : Char
1947: Function nextToken : WideString
1948: function NextToken:Char
1949: Function Norm( const Data : array of Double) : Extended
1950: Function NormalizeAngle( const Angle : Extended) : Extended
1951: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
1952: Function NormalizeRect( const Rect : TRect) : TRect
1953: function NormalizeRect(const Rect: TRect): TRect;
1954: Function Now : TDateTime
1955: function Now2: tDateTime
1956: Function NumProcessThreads : integer
1957: Function NumThreadCount : integer
1958: Function NthDayOfWeek( const AValue : TDateTime) : Word
1959: Function NtProductType : TNtProductType
1960: Function NtProductTypeString : string
1961: function Null: Variant;
1962: Function NullPoint : TPoint
1963: Function NullRect : TRect
1964: Function Null2Blank(aString:String):String;
1965: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
1966: Function NumIP : integer
1967: function Odd(x: Longint): boolean;
1968: Function OffsetFromUTC : TDateTime
1969: Function OffsetPoint( const P, Offset : TPoint) : TPoint
1970: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
1971: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
1972: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
1973: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
1974: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
1975: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
1976: function OpenBit:Integer
1977: Function OpenDatabase : TDatabase
1978: Function OpenDatabase( const DatabaseName : string) : TDatabase
1979: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
1980: Function OpenObject( Value : PChar) : Boolean;
1981: Function OpenObject1( Value : string) : Boolean;
1982: Function OpenSession( const SessionName : string) : TSession
1983: Function OpenVolume( const Drive : Char) : THandle
1984: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
1985: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
1986: Function OrdToBinary( const Value : Byte) : string;
1987: Function OrdToBinary1( const Value : Shortint) : string;
1988: Function OrdToBinary2( const Value : Smallint) : string;
1989: Function OrdToBinary3( const Value : Word) : string;
1990: Function OrdToBinary4( const Value : Integer) : string;
1991: Function OrdToBinary5( const Value : Cardinal) : string;
1992: Function OrdToBinary6( const Value : Int64) : string;
1993: Function OSCheck( RetVal : Boolean) : Boolean
1994: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
1995: Function OSIdentToString( const OSIdent : DWORD) : string
1996: Function Output: Text)
1997: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
1998: Function Owner : TCustomListView
1999: function Owner : TPersistent
2000: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
2001: Function PadL( pStr : String; pLth : integer) : String
2002: Function Padl(s : AnyString;I : longInt) : AnyString;
2003: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
2004: Function PadR( pStr : String; pLth : integer) : String
2005: Function Padr(s : AnyString;I : longInt) : AnyString;
2006: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
2007: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2008: Function Padz(s : AnyString;I : longInt) : AnyString;
```

```
2009: Function PaethPredictor( a, b, c : Byte) : Byte
2010: Function PARAMBYNAME( const VALUE : String) : TPARAM
2011: Function ParamByName( const Value : WideString) : TParameter
2012: Function ParamCount: Integer
2013: Function ParamsEncode( const ASrc : string) : string
2014: function ParamStr(Index: Integer): string)
2015: Function ParseDate( const DateStr : string) : TDateTime
2016: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2017: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2018: Function PathAddExtension( const Path, Extension : string) : string
2019: Function PathAddSeparator( const Path : string) : string
2020: Function PathAppend( const Path, Append : string) : string
2021: Function PathBuildRoot( const Drive : Byte) : string
2022: Function PathCanonicalize( const Path : string) : string
2023: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2024: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2025: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2026: Function PathEncode( const ASrc : string) : string
2027: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2028: Function PathExtractFileNameNoExt( const Path : string) : string
2029: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2030: Function PathGetDepth( const Path : string) : Integer
2031: Function PathGetLongName( const Path : string) : string
2032: Function PathGetLongName2( Path : string) : string
2033: Function PathGetShortName( const Path : string) : string
2034: Function PathIsAbsolute( const Path : string) : Boolean
2035: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2036: Function PathIsDiskDevice( const Path : string) : Boolean
2037: Function PathIsUNC( const Path : string) : Boolean
2038: Function PathRemoveExtension( const Path : string) : string
2039: Function PathRemoveSeparator( const Path : string) : string
2040: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2041: Function Peek : Pointer
2042: Function Peek : TObject
2043: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2044: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2045: function Permutation(npr, k: integer): extended;
2046: function PermutationInt(npr, k: integer): Int64;
2047: Function PermutationJ( N, R : Cardinal) : Float
2048: Function Pi : Extended;
2049: Function PiE : Extended;
2050: Function PixelsToDialogsX( const Pixels : Word) : Word
2051: Function PixelsToDialogsY( const Pixels : Word) : Word
2052: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2053: Function Point( X, Y : Integer) : TPoint
2054: function Point(X, Y: Integer): TPoint)
2055: Function PointAssign( const X, Y : Integer) : TPoint
2056: Function PointDist( const P1, P2 : TPoint) : Double;
2057: function PointDist(const P1,P2: TFloatPoint): Double;
2058: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2059: function PointDist2(const P1,P2: TPoint): Double;
2060: Function PointEqual( const P1, P2 : TPoint) : Boolean
2061: Function PointIsNull( const P : TPoint) : Boolean
2062: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2063: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2064: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2065: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2066: Function Pop : Pointer
2067: Function Pop : TObject
2068: Function PopnStdDev( const Data : array of Double) : Extended
2069: Function PopnVariance( const Data : array of Double) : Extended
2070: Function PopulationVariance( const X : TDynFloatArray) : Float
2071: function Pos(SubStr, S: AnyString): Longint;
2072: Function PosEqual( const Rect : TRect) : Boolean
2073: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2074: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2075: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2076: Function Post1( AURL : string; const ASource : TStrings) : string;
2077: Function Post2( AURL : string; const ASource : TStream) : string;
2078: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2079: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2080: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2081: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2082: Function Power( const Base, Exponent : Extended) : Extended
2083: Function PowerBig(aval, n:integer): string;
2084: Function PowerIntJ( const X : Float; N : Integer) : Float;
2085: Function PowerJ( const Base, Exponent : Float) : Float;
2086: Function PowerOffOS : Boolean
2087: Function PreformatDateString( Ps : string) : string
2088: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2089: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2090: Function Printer : TPrinter
2091: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2092: Function ProcessResponse : TIdHTTPWhatsNext
2093: Function ProduceContent : string
2094: Function ProduceContentFromStream( Stream : TStream) : string
```

```
2095: Function ProduceContentFromString( const S : string) : string
2096: Function ProgIDToClassID(const ProgID: string): TGUID;
2097: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2098: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2099: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2100: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2101: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2102: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2103: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2104: Function Push( AItem : Pointer) : Pointer
2105: Function Push( AObject : TObject) : TObject
2106: Function Put1( AURL : string; const ASource : TStream) : string;
2107: Function Pythagoras( const X, Y : Extended) : Extended
2108: Function queryDLLInterface( var queryList : TStringList) : TStringList
2109: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2110: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2111: Function queryPerformanceCounter2(mse: int64): int64;
2112: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2113: //Function QueryPerformanceFrequency(mse: int64): boolean;
2114: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2115: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2116: Procedure QueryPerformanceCounter1(var aC: Int64);
2117: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2118: Function Quote( const ACommand : String) : SmallInt
2119: Function QuotedStr( S : string) : string
2120: Function RadToCycle( const Radians : Extended) : Extended
2121: Function RadToDeg( const Radians : Extended) : Extended
2122: Function RadToDeg( const Value : Extended) : Extended;
2123: Function RadToDeg1( const Value : Double) : Double;
2124: Function RadToDeg2( const Value : Single) : Single;
2125: Function RadToGrad( const Radians : Extended) : Extended
2126: Function RadToGrad( const Value : Extended) : Extended;
2127: Function RadToGrad1( const Value : Double) : Double;
2128: Function RadToGrad2( const Value : Single) : Single;
2129: Function RandG( Mean, StdDev : Extended) : Extended
2130: function Random(const ARange: Integer): Integer;
2131: function random2(a: integer): double
2132: function RandomE: Extended;
2133: function RandomF: Extended;
2134: Function RandomFrom( const AValues : array of string) : string;
2135: Function RandomRange( const AFrom, ATo : Integer) : Integer
2136: function randSeed: longint
2137: Function RawToDataColumn( ACol : Integer) : Integer
2138: Function Read : Char
2139: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2140: function Read(Buffer:String;Count:LongInt):LongInt
2141: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2142: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2143: Function ReadCardinal( const AConvert : boolean) : Cardinal
2144: Function ReadChar : Char
2145: Function ReadClient( var Buffer, Count : Integer) : Integer
2146: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2147: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2148: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2149: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
      Boolean):Integer
2150: Function ReadInteger( const AConvert : boolean) : Integer
2151: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2152: Function ReadLn : string
2153: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2154: function Readln(question: string): string;
2155: Function ReadLnWait( AFailCount : Integer) : string
2156: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2157: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2158: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2159: Function ReadString( const ABytes : Integer) : string
2160: Function ReadString( const Section, Ident, Default : string) : string
2161: Function ReadString( Count : Integer) : string
2162: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2163: Function ReadTimeStampCounter : Int64
2164: Function RebootOS : Boolean
2165: Function Receive( ATimeOut : Integer) : TReplyStatus
2166: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2167: Function ReceiveLength : Integer
2168: Function ReceiveText : string
2169: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2170: Function ReceiveSerialText: string
2171: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2172: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
      AMilliSec:Word):TDateTime
2173: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2174: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2175: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2176: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2177: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2178: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2179: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
```

```
2180: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2181: Function Reconcile( const Results : OleVariant) : Boolean
2182: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2183: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2184: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2185: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2186: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2187: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2188: Function RectCenter( const R : TRect) : TPoint
2189: Function RectEqual( const R1, R2 : TRect) : Boolean
2190: Function RectHeight( const R : TRect) : Integer
2191: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2192: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2193: Function RectIntersection( const R1, R2 : TRect) : TRect
2194: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2195: Function RectIsEmpty( const R : TRect) : Boolean
2196: Function RectIsNull( const R : TRect) : Boolean
2197: Function RectIsSquare( const R : TRect) : Boolean
2198: Function RectIsValid( const R : TRect) : Boolean
2199: Function RectsAreValid( R : array of TRect) : Boolean
2200: Function RectUnion( const R1, R2 : TRect) : TRect
2201: Function RectWidth( const R : TRect) : Integer
2202: Function RedComponent( const Color32 : TColor32) : Integer
2203: Function Refresh : Boolean
2204: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2205: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2206: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2207: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2208: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2209: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2210: Function ReleaseHandle : HBITMAP
2211: Function ReleaseHandle : HENHMETAFILE
2212: Function ReleaseHandle : HICON
2213: Function ReleasePalette : HPALETTE
2214: Function RemainderFloat( const X, Y : Float) : Float
2215: Function Remove( AClass : TClass) : Integer
2216: Function Remove( AComponent : TComponent) : Integer
2217: Function Remove( AItem : Integer) : Integer
2218: Function Remove( AItem : Pointer) : Pointer
2219: Function Remove( AItem : TObject) : TObject
2220: Function Remove( AObject : TObject) : Integer
2221: Function RemoveBackslash( const PathName : string) : string
2222: Function RemoveDF( aString : String) : String  //removes thousand separator
2223: Function RemoveDir( Dir : string) : Boolean
2224: function RemoveDir(const Dir: string): Boolean)
2225: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2226: Function RemoveFileExt( const FileName : string) : string
2227: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2228: Function RenameFile( OldName, NewName : string) : Boolean
2229: function RenameFile(const OldName: string; const NewName: string): Boolean)
2230: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2231: Function ReplaceText( const AText, AFromText, AToText : string) : string
2232: Function Replicate(c : char;I : longInt) : String;
2233: Function Request : TWebRequest
2234: Function ResemblesText( const AText, AOther : string) : Boolean
2235: Function Reset : Boolean
2236: function Reset2(mypath: string):string;
2237: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2238: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2239: Function Response : TWebResponse
2240: Function ResumeSupported : Boolean
2241: Function RETHINKHOTKEYS : BOOLEAN
2242: Function RETHINKLINES : BOOLEAN
2243: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2244: Function RetrieveCurrentDir : string
2245: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2246: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2247: Function RetrieveMailBoxSize : integer
2248: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2249: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2250: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2251: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2252: Function ReverseBits( Value : Byte) : Byte;
2253: Function ReverseBits1( Value : Shortint) : Shortint;
2254: Function ReverseBits2( Value : Smallint) : Smallint;
2255: Function ReverseBits3( Value : Word) : Word;
2256: Function ReverseBits4( Value : Cardinal) : Cardinal;
2257: Function ReverseBits4( Value : Integer) : Integer;
2258: Function ReverseBits5( Value : Int64) : Int64;
2259: Function ReverseBytes( Value : Word) : Word;
2260: Function ReverseBytes1( Value : Smallint) : Smallint;
2261: Function ReverseBytes2( Value : Integer) : Integer;
2262: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2263: Function ReverseBytes4( Value : Int64) : Int64;
2264: Function ReverseString( const AText : string) : string
2265: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var
      HostName:String):Bool;
2266: Function Revert : HResult
2267: Function RGB(R,G,B: Byte): TColor;
```

```
2268: Function RGB2BGR( const Color : TColor) : TColor
2269: Function RGB2TColor( R, G, B : Byte) : TColor
2270: Function RGBToWebColorName( RGB : Integer) : string
2271: Function RGBToWebColorStr( RGB : Integer) : string
2272: Function RgbToHtml( Value : TColor) : string
2273: Function HtmlToRgb(const Value: string): TColor;
2274: Function RightStr( const AStr : String; Len : Integer) : String
2275: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2276: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2277: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2278: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2279: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2280: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2281: Function Round(e : Extended) : Longint;
2282: Function Round64(e: extended): Int64;
2283: Function RoundAt( const Value : string; Position : SmallInt) : string
2284: Function RoundFrequency( const Frequency : Integer) : Integer
2285: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2286: Function RoundPoint( const X, Y : Double) : TPoint
2287: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2288: Function RowCount : Integer
2289: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2290: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2291: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2292: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2293: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2294: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2295: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2296: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2297: Function S_AddBackSlash( const ADirName : string) : string
2298: Function S_AllTrim( const cStr : string) : string
2299: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2300: Function S_Cut( const cStr : string; const iLen : integer) : string
2301: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2302: Function S_DirExists( const ADir : string) : Boolean
2303: Function S_Empty( const cStr : string) : boolean
2304: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2305: Function S_LargeFontsActive : Boolean
2306: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2307: Function S_LTrim( const cStr : string) : string
2308: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2309: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2310: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2311: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2312: Function S_RTrim( const cStr : string) : string
2313: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2314: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2315: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2316: Function S_Space( const iLen : integer) : String
2317: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2318: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2319: Function S_StrCRC32( const Text : string) : LongWORD
2320: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2321: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2322: Function S_StringtoUTF_8( const AString : string) : string
2323: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2324: function S_StrToReal(const cStr: string; var R: Double): Boolean
2325: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2326: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2327: Function S_UTF_8ToString( const AString : string) : string
2328: Function S_WBox( const AText : string) : integer
2329: Function SameDate( const A, B : TDateTime) : Boolean
2330: function SameDate(const A, B: TDateTime): Boolean;
2331: Function SameDateTime( const A, B : TDateTime) : Boolean
2332: function SameDateTime(const A, B: TDateTime): Boolean;
2333: Function SameFileName( S1, S2 : string) : Boolean
2334: Function SameText( S1, S2 : string) : Boolean
2335: function SameText(const S1: string; const S2: string): Boolean)
2336: Function SameTime( const A, B : TDateTime) : Boolean
2337: function SameTime(const A, B: TDateTime): Boolean;
2338: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2339: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2340: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2341: Function SampleVariance( const X : TDynFloatArray) : Float
2342: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2343: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2344: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2345: Function SaveToFile( const AFileName : TFileName) : Boolean
2346: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2347: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2348: Function ScanF(const aformat: String; const args: array of const): string;
2349: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2350: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2351: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2352: function SearchRecattr: integer;
2353: function SearchRecExcludeAttr: integer;
2354: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
```

```
2355: function SearchRecname: string;
2356: function SearchRecsize: integer;
2357: function SearchRecTime: integer;
2358: Function Sec( const X : Extended) : Extended
2359: Function Secant( const X : Extended) : Extended
2360: Function SecH( const X : Extended) : Extended
2361: Function SecondOf( const AValue : TDateTime) : Word
2362: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2363: Function SecondOfTheHour( const AValue : TDateTime) : Word
2364: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2365: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2366: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2367: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2368: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2369: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2370: Function SectionExists( const Section : string) : Boolean
2371: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2372: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2373: function Seek(Offset:LongInt;Origin:Word):LongInt
2374: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2375: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2376: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2377: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2378: Function SendBuf( var Buf, Count : Integer) : Integer
2379: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2380: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2381: Function SendKey( AppName : string; Key : Char) : Boolean
2382: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2383: Function SendStream( AStream : TStream) : Boolean
2384: Function SendStreamThenDrop( AStream : TStream) : Boolean
2385: Function SendText( const S : string) : Integer
2386: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2387: Function SendSerialText(Data: String): cardinal
2388: Function Sent : Boolean
2389: Function ServicesFilePath: string
2390: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2391: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2392: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2393: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2394: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2395: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2396: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2397: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2398: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2399: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2400: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2401: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2402: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2403: Function SetCurrentDir( Dir : string) : Boolean
2404: function SetCurrentDir(const Dir: string): Boolean)
2405: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2406: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2407: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2408: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2409: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2410: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2411: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2412: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2413: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2414: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2415: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2416: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2417: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2418: Function SetLocalTime( Value : TDateTime) : boolean
2419: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2420: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2421: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2422: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2423: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2424: Function SetSize( libNewSize : Longint) : HResult
2425: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2426: Function Sgn( const X : Extended) : Integer
2427: function SHA1(const fileName: string): string;
2428: function SHA256(astr: string; amode: char): string)
2429: function SHA512(astr: string; amode: char): string)
2430: Function ShareMemoryManager : Boolean
2431: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2432: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2433: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2434: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2435: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2436: function ShortDateFormat: string;
2437: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2438: function ShortTimeFormat: string;
2439: function SHOWMODAL:INTEGER
2440: Function ShowModalControl(aControl : TControl; BS : TFormBorderStyle; BI : TBorderIcons; WS :
      TWindowState; aColor : TColor; BW : Integer; Title : String; BeforeShowModal : TNotifyEvent) :
      TModalResult');
```

```
2441: Function
      ShowModalPanel(aPnl:TCustomPanel;Titl:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2442: function ShowWindow(C1: HWND; C2: integer): boolean;
2443: procedure ShowMemory     //in Dialog
2444: function ShowMemory2: string;
2445: Function ShutDownOS : Boolean
2446: Function Signe( const X, Y : Extended) : Extended
2447: Function Sign( const X : Extended) : Integer
2448: Function Sin(e : Extended) : Extended;
2449: Function sinc( const x : Double) : Double
2450: Function SinJ( X : Float) : Float
2451: Function Size( const AFileName : String) : Integer
2452: function SizeOf: Longint;
2453: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2454: function SlashSep(const Path, S: String): String
2455: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2456: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2457: Function SmallPoint(X, Y: Integer): TSmallPoint)
2458: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2459: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2460: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2461: Function SoundexProc( const AText, AOther : string) : Boolean
2462: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2463: Function SoundexWord( const AText : string) : Word
2464: Function SourcePos : Longint
2465: function SourcePos:LongInt
2466: Function Split0( Str : string; const substr : string) : TStringList
2467: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2468: Function SQLRequiresParams( const SQL : WideString) : Boolean
2469: Function Sqr(e : Extended) : Extended;
2470: Function Sqrt(e : Extended) : Extended;
2471: Function StartIP : String
2472: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2473: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2474: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2475: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2476: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2477: Function StartOfAYear( const AYear : Word) : TDateTime
2478: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2479: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2480: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2481: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2482: Function StartsStr( const ASubText, AText : string) : Boolean
2483: Function StartsText( const ASubText, AText : string) : Boolean
2484: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2485: Function StartsWith( const str : string; const sub : string) : Boolean
2486: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2487: Function StatusString( StatusCode : Integer) : string
2488: Function StdDev( const Data : array of Double) : Extended
2489: Function Stop : Float
2490: Function StopCount( var Counter : TJclCounter) : Float
2491: Function StoreColumns : Boolean
2492: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2493: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2494: Function StrAlloc( Size : Cardinal) : PChar
2495: function StrAlloc(Size: Cardinal): PChar)
2496: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2497: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2498: Function StrBufSize( Str : PChar) : Cardinal
2499: function StrBufSize(const Str: PChar): Cardinal)
2500: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2501: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2502: Function StrCat( Dest : PChar; Source : PChar) : PChar
2503: function StrCat(Dest: PChar; const Source: PChar): PChar)
2504: Function StrCharLength( Str : PChar) : Integer
2505: Function StrComp( Str1, Str2 : PChar) : Integer
2506: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2507: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2508: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2509: Function Stream_to_AnsiString( Source : TStream) : ansistring
2510: Function Stream_to_Base64( Source : TStream) : ansistring
2511: Function Stream_to_decimalbytes( Source : TStream) : string
2512: Function Stream2WideString( oStream : TStream) : WideString
2513: Function StreamtoAnsiString( Source : TStream) : ansistring
2514: Function StreamToByte( Source : TStream) : string
2515: Function StreamToDecimalbytes( Source : TStream) : string
2516: Function StreamtoOrd( Source : TStream) : string
2517: Function StreamToString( Source : TStream) : string
2518: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2519: Function StrEmpty( const sString : string) : boolean
2520: Function StrEnd( Str : PChar) : PChar
2521: function StrEnd(const Str: PChar): PChar)
2522: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2523: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2524: Function StrGet(var S : String; I : Integer) : Char;
2525: Function StrGet2(S : String; I : Integer) : Char;
2526: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2527: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2528: Function StrHtmlDecode( const AStr : String) : String
```

```
2529: Function StrHtmlEncode( const AStr : String) : String
2530: Function StrToBytes(const Value: String): TBytes;
2531: Function StrIComp( Str1, Str2 : PChar) : Integer
2532: Function StringOfChar(c : char;I : longInt) : String;
2533: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2534: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2535: Function StringRefCount(const s: String): integer;
2536: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2537: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2538: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2539: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2540: Function StringToBoolean( const Ps : string) : Boolean
2541: function StringToColor(const S: string): TColor)
2542: function StringToCursor(const S: string): TCursor;
2543: function StringToGUID(const S: string): TGUID)
2544: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2545: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2546: Function StringWidth( S : string) : Integer
2547: Function StrInternetToDateTime( Value : string) : TDateTime
2548: Function StrIsDateTime( const Ps : string) : Boolean
2549: Function StrIsFloatMoney( const Ps : string) : Boolean
2550: Function StrIsInteger( const S : string) : Boolean
2551: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2552: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2553: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2554: Function StrLen( Str : PChar) : Cardinal
2555: function StrLen(const Str: PChar): Cardinal)
2556: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2557: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2558: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2559: Function StrLower( Str : PChar) : PChar
2560: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2561: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2562: Function StrNew( Str : PChar) : PChar
2563: function StrNew(const Str: PChar): PChar)
2564: Function StrNextChar( Str : PChar) : PChar
2565: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2566: Function StrParse( var sString : string; const sDelimiters : string) : string;
2567: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2568: Function StrPas( Str : PChar) : string
2569: function StrPas(const Str: PChar): string)
2570: Function StrPCopy( Dest : PChar; Source : string) : PChar
2571: function StrPCopy(Dest: PChar; const Source: string): PChar)
2572: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2573: Function StrPos( Str1, Str2 : PChar) : PChar
2574: Function StrScan(const Str: PChar; Chr: Char): PChar)
2575: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2576: Function StrToBcd( const AValue : string) : TBcd
2577: Function StrToBool( S : string) : Boolean
2578: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2579: Function StrToCard( const AStr : String) : Cardinal
2580: Function StrToConv( AText : string; out AType : TConvType) : Double
2581: Function StrToCurr( S : string) : Currency;
2582: function StrToCurr(const S: string): Currency)
2583: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2584: Function StrToDate( S : string) : TDateTime;
2585: function StrToDate(const s: string): TDateTime;
2586: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2587: Function StrToDateTime( S : string) : TDateTime;
2588: function StrToDateTime(const S: string): TDateTime)
2589: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2590: Function StrToDay( const ADay : string) : Byte
2591: Function StrToFloat( S : string) : Extended;
2592: function StrToFloat(s: String): Extended;
2593: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2594: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2595: Function StrToFloat( S : string) : Extended;
2596: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2597: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2598: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2599: Function StrToCurr( S : string) : Currency;
2600: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2601: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2602: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2603: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2604: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2605: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2606: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2607: Function StrToDateTime( S : string) : TDateTime;
2608: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2609: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2610: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2611: Function StrToInt( S : string) : Integer
2612: function StrToInt(s: String): Longint;
2613: Function StrToInt64( S : string) : Int64
2614: function StrToInt64(s: String): int64;
2615: Function StrToInt64Def( S : string; Default : Int64) : Int64
2616: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2617: Function StrToIntDef( S : string; Default : Integer) : Integer
```

```
2618: function StrToIntDef(const S: string; Default: Integer): Integer)
2619: function StrToIntDef(s: String; def: Longint): Longint;
2620: Function StrToMonth( const AMonth : string) : Byte
2621: Function StrToTime( S : string) : TDateTime;
2622: function StrToTime(const S: string): TDateTime)
2623: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2624: Function StrToWord( const Value : String) : Word
2625: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2626: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2627: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2628: Function StrUpper( Str : PChar) : PChar
2629: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2630: Function Sum( const Data : array of Double) : Extended
2631: Function SumFloatArray( const B : TDynFloatArray) : Float
2632: Function SumInt( const Data : array of Integer) : Integer
2633: Function SumOfSquares( const Data : array of Double) : Extended
2634: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2635: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2636: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2637: Function Supports( CursorOptions : TCursorOptions) : Boolean
2638: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2639: Function SwapWord(w : word): word)
2640: Function SwapInt(i : integer): integer)
2641: Function SwapLong(L : longint): longint)
2642: Function Swap(i : integer): integer)
2643: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2644: Function SyncTime : Boolean
2645: Function SysErrorMessage( ErrorCode : Integer) : string
2646: function SysErrorMessage(ErrorCode: Integer): string)
2647: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2648: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2649: Function SysStringLen(const S: WideString): Integer; stdcall;
2650: Function TabRect( Index : Integer) : TRect
2651: Function Tan( const X : Extended) : Extended
2652: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2653: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2654: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2655: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2656: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2657: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2658: Function TenToY( const Y : Float) : Float
2659: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2660: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2661: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2662: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2663: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2664: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2665: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2666: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2667: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2668: Function TestBits( const Value, Mask : Byte) : Boolean;
2669: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2670: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2671: Function TestBits3( const Value, Mask : Word) : Boolean;
2672: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2673: Function TestBits5( const Value, Mask : Integer) : Boolean;
2674: Function TestBits6( const Value, Mask : Int64) : Boolean;
2675: Function TestFDIVInstruction : Boolean
2676: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2677: Function TextExtent( const Text : string) : TSize
2678: function TextHeight(Text: string): Integer;
2679: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2680: Function TextStartsWith( const S, SubS : string) : Boolean
2681: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2682: Function ConvInteger(i : integer):string;
2683: Function IntegerToText(i : integer):string;
2684: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2685: function TextWidth(Text: string): Integer;
2686: Function ThreadCount : integer
2687: function ThousandSeparator: char;
2688: Function Ticks : Cardinal
2689: Function Time : TDateTime
2690: function Time: TDateTime;
2691: function TimeGetTime: int64;
2692: Function TimeOf( const AValue : TDateTime) : TDateTime
2693: function TimeSeparator: char;
2694: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2695: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2696: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2697: Function TimeToStr( DateTime : TDateTime) : string;
2698: function TimeToStr(const DateTime: TDateTime): string;
2699: Function TimeZoneBias : TDateTime
2700: Function ToCommon( const AValue : Double) : Double
```

```
2701: function ToCommon(const AValue: Double): Double;
2702: Function Today : TDateTime
2703: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2704: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2705: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2706: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2707: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2708: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2709: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2710: function TokenComponentIdent:String
2711: Function TokenFloat : Extended
2712: function TokenFloat:Extended
2713: Function TokenInt : Longint
2714: function TokenInt:LongInt
2715: Function TokenString : string
2716: function TokenString:String
2717: Function TokenSymbolIs( const S : string) : Boolean
2718: function TokenSymbolIs(S:String):Boolean
2719: Function Tomorrow : TDateTime
2720: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2721: Function ToString : string
2722: Function TotalVariance( const Data : array of Double) : Extended
2723: Function Trace2( AURL : string) : string;
2724: Function TrackMenu( Button : TToolButton) : Boolean
2725: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2726: Function TranslateURI( const URI : string) : string
2727: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2728: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2729: Function Trim( S : string) : string;
2730: Function Trim( S : WideString) : WideString;
2731: Function Trim(s : AnyString) : AnyString;
2732: Function TrimAllOf( ATrim, AText : String) : String
2733: Function TrimLeft( S : string) : string;
2734: Function TrimLeft( S : WideString) : WideString;
2735: function TrimLeft(const S: string): string)
2736: Function TrimRight( S : string) : string;
2737: Function TrimRight( S : WideString) : WideString;
2738: function TrimRight(const S: string): string)
2739: function TrueBoolStrs: array of string
2740: Function Trunc(e : Extended) : Longint;
2741: Function Trunc64(e: extended): Int64;
2742: Function TruncPower( const Base, Exponent : Float) : Float
2743: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2744: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2745: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2746: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2747: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2748: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
      AValue:TDateTime):Boolean
2749: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2750: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
      AVal:TDateTime):Bool
2751: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2752: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2753: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2754: Function TryLock : Boolean
2755: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2756: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
2757: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2758: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2759: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2760: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2761: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2762: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2763: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2764: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2765: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2766: Function TwoToY( const Y : Float) : Float
2767: Function UCS4StringToWideString( const S : UCS4String) : WideString
2768: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2769: function Unassigned: Variant;
2770: Function UndoLastChange( FollowChange : Boolean) : Boolean
2771: function UniCodeToStr(Value: string): string;
2772: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2773: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2774: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2775: Function UnixPathToDosPath( const Path : string) : string
2776: Function UnixToDateTime( const AValue : Int64) : TDateTime
2777: function UnixToDateTime(U: Int64): TDateTime;
2778: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2779: Function UnlockResource( ResData : HGLOBAL) : LongBool
2780: Function UnlockVolume( var Handle : THandle) : Boolean
2781: Function UnMaskString( Mask, Value : String) : String
2782: function UpCase(ch : Char ) : Char;
2783: Function UpCaseFirst( const AStr : string) : string
2784: Function UpCaseFirstWord( const AStr : string) : string
2785: Function UpdateAction( Action : TBasicAction) : Boolean
```

```
2786: Function UpdateKind : TUpdateKind
2787: Function UPDATESTATUS : TUPDATESTATUS
2788: Function UpperCase( S : string) : string
2789: Function Uppercase(s : AnyString) : AnyString;
2790: Function URLDecode( ASrc : string) : string
2791: Function URLEncode( const ASrc : string) : string
2792: Function UseRightToLeftAlignment : Boolean
2793: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2794: Function UseRightToLeftReading : Boolean
2795: Function UTF8CharLength( Lead : Char) : Integer
2796: Function UTF8CharSize( Lead : Char) : Integer
2797: Function UTF8Decode( const S : UTF8String) : WideString
2798: Function UTF8Encode( const WS : WideString) : UTF8String
2799: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2800: Function Utf8ToAnsi( const S : UTF8String) : string
2801: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2802: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2803: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2804: Function ValidParentForm(control: TControl): TForm
2805: Function Value : Variant
2806: Function ValueExists( const Section, Ident : string) : Boolean
2807: Function ValueOf( const Key : string) : Integer
2808: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2809: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2810: Function VarArrayFromStrings( Strings : TStrings) : Variant
2811: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2812: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2813: Function VarFMTBcd : TVarType
2814: Function VarFMTBcdCreate1 : Variant;
2815: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2816: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2817: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2818: Function Variance( const Data : array of Double) : Extended
2819: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2820: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2821: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2822: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2823: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2824: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2825: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2826: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2827: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2828: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2829: Function VariantNeg( const V1 : Variant) : Variant
2830: Function VariantNot( const V1 : Variant) : Variant
2831: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2832: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2833: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2834: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2835: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2836: function VarIsEmpty(const V: Variant): Boolean;
2837: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2838: function VarIsNull(const V: Variant): Boolean;
2839: Function VarToBcd( const AValue : Variant) : TBcd
2840: function VarType(const V: Variant): TVarType;
2841: Function VarType( const V : Variant) : TVarType
2842: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2843: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2844: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2845: Function VarIsByRef( const V : Variant) : Boolean
2846: Function VarIsEmpty( const V : Variant) : Boolean
2847: Procedure VarCheckEmpty( const V : Variant)
2848: Function VarIsNull( const V : Variant) : Boolean
2849: Function VarIsClear( const V : Variant) : Boolean
2850: Function VarIsCustom( const V : Variant) : Boolean
2851: Function VarIsOrdinal( const V : Variant) : Boolean
2852: Function VarIsFloat( const V : Variant) : Boolean
2853: Function VarIsNumeric( const V : Variant) : Boolean
2854: Function VarIsStr( const V : Variant) : Boolean
2855: Function VarToStr( const V : Variant) : string
2856: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2857: Function VarToWideStr( const V : Variant) : WideString
2858: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2859: Function VarToDateTime( const V : Variant) : TDateTime
2860: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2861: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2862: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2863: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2864: Function VarSameValue( const A, B : Variant) : Boolean
2865: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2866: Function VarIsEmptyParam( const V : Variant) : Boolean
2867: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2868: Function VarIsError1( const V : Variant) : Boolean;
2869: Function VarAsError( AResult : HRESULT) : Variant
2870: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2871: Function VarIsArray( const A : Variant) : Boolean;
2872: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2873: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2874: Function VarArrayOf( const Values : array of Variant) : Variant
```

```
2875: Function VarArrayRef( const A : Variant) : Variant
2876: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2877: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2878: Function VarArrayDimCount( const A : Variant) : Integer
2879: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2880: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2881: Function VarArrayLock( const A : Variant) : ___Pointer
2882: Procedure VarArrayUnlock( const A : Variant)
2883: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2884: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2885: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2886: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2887: Function Unassigned : Variant
2888: Function Null : Variant
2889: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2890: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2891: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2892: function VectorDot(const V1,V2: TFloatPoint): Double;
2893: Function VectorLengthSqr( const V : TFloatPoint) : Double
2894: function VectorLengthSqr(const V: TFloatPoint): Double;
2895: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2896: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2897: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2898: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2899: Function Verify( AUserName : String) : String
2900: Function Versine( X : Float) : Float
2901: function VersionCheck: boolean;
2902: function VersionCheckAct: string;
2903: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2904: Function VersionLanguageName( const LangId : Word) : string
2905: Function VersionResourceAvailable( const FileName : string) : Boolean
2906: Function Visible : Boolean
2907: function VolumeID(DriveChar: Char): string
2908: Function WaitFor( const AString : string) : string
2909: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2910: Function WaitFor1 : TWaitResult;
2911: Function WaitForData( Timeout : Longint) : Boolean
2912: Function WebColorNameToColor( WebColorName : string) : TColor
2913: Function WebColorStrToColor( WebColor : string) : TColor
2914: Function WebColorToRGB( WebColor : Integer) : Integer
2915: Function wGet(aURL, afile: string): boolean;'
2916: Function wGet2(aURL, afile: string): boolean;'  //without file open
2917: Function WebGet(aURL, afile: string): boolean;'
2918: Function WebExists: boolean;   //alias to isinternet
2919: Function WeekOf( const AValue : TDateTime) : Word
2920: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2921: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2922: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2923: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2924: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2925: Function WeeksInAYear( const AYear : Word) : Word
2926: Function WeeksInYear( const AValue : TDateTime) : Word
2927: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2928: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2929: Function WideCat( const x, y : WideString) : WideString
2930: Function WideCompareStr( S1, S2 : WideString) : Integer
2931: function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
2932: Function WideCompareText( S1, S2 : WideString) : Integer
2933: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
2934: Function WideCopy( const src : WideString; index, count : Integer) : WideString
2935: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
2936: Function WideEqual( const x, y : WideString) : Boolean
2937: Function WideFormat(const Format: WideString; const Args: array of const): WideString)
2938: Function WideGreater( const x, y : WideString) : Boolean
2939: Function WideLength( const src : WideString) : Integer
2940: Function WideLess( const x, y : WideString) : Boolean
2941: Function WideLowerCase( S : WideString) : WideString
2942: function WideLowerCase(const S: WideString): WideString)
2943: Function WidePos( const src, sub : WideString) : Integer
2944: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
2945: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
2946: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
2947: Function WideSameStr( S1, S2 : WideString) : Boolean
2948: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
2949: Function WideSameText( S1, S2 : WideString) : Boolean
2950: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
2951: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
2952: Function WideStringToUCS4String( const S : WideString) : UCS4String
2953: Function WideUpperCase( S : WideString) : WideString
2954: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
2955: function Win32Check(RetVal: boolean): boolean)
2956: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
2957: Function Win32RestoreFile( const FileName : string) : Boolean
2958: Function Win32Type : TIdWin32Type
2959: Function WinColor( const Color32 : TColor32) : TColor
2960: function winexec(FileName: pchar; showCmd: integer): integer;
2961: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
2962: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
2963: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
```

```
2964: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
2965: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
2966: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
2967: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
2968: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
2969: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
2970: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
2971: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
2972: Function WordToStr( const Value : Word) : String
2973: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
2974: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
2975: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
2976: Function WorkArea : Integer
2977: Function WrapText( Line : string; MaxCol : Integer) : string;
2978: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
2979: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
2980: function Write(Buffer:String;Count:LongInt):LongInt
2981: Function WriteClient( var Buffer, Count : Integer) : Integer
2982: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
2983: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
2984: Function WriteString( const AString : string) : Boolean
2985: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
2986: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
2987: Function wsprintf( Output : PChar; Format : PChar) : Integer
2988: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
2989: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
2990: Function XorDecode( const Key, Source : string) : string
2991: Function XorEncode( const Key, Source : string) : string
2992: Function XorString( const Key, Src : ShortString) : ShortString
2993: Function Yield : Bool
2994: Function YearOf( const AValue : TDateTime) : Word
2995: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
2996: Function YearSpan( const ANow, AThen : TDateTime) : Double
2997: Function Yesterday : TDateTime
2998: Function YesNoDialog(const ACaption, AMsg: string): boolean;
2999: Function( const Name : string; Proc : TUserFunction)
3000: Function using Special_Scholz from 3.8.5.0
3001: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3002: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3003: Function FloatToTime2Dec(value:Extended):Extended;
3004: Function MinToStd(value:Extended):Extended;
3005: Function MinToStdAsString(value:Extended):String;
3006: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3007: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3008: Function Round2Dec (zahl:Extended):Extended;
3009: Function GetAngle(x,y:Extended):Double;
3010: Function AddAngle(a1,a2:Double):Double;
3011:
3012: ************************************************************************
3013: unit uPSI_StText;
3014: ************************************************************************
3015: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3016: Function TextFileSize( var F : TextFile) : LongInt
3017: Function TextPos( var F : TextFile) : LongInt
3018: Function TextFlush( var F : TextFile) : Boolean
3019:
3020: ************************************************************************
3021: from  JvVCLUtils;
3022: ************************************************************************
3023: { Windows resources (bitmaps and icons) VCL-oriented routines }
3024: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3025: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
     DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3026: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
     Bitmap: TBitmap; TransparentColor: TColor);
3027: function MakeBitmap(ResID: PChar): TBitmap;
3028: function MakeBitmapID(ResID: Word): TBitmap;
3029: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3030: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3031: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3032: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3033:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3034: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3035: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3036: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3037: {$IFDEF WIN32}
3038: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3039:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3040: {$ENDIF}
3041: function MakeIcon(ResID: PChar): TIcon;
3042: function MakeIconID(ResID: Word): TIcon;
3043: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3044: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3045: {$IFDEF WIN32}
3046: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3047: {$ENDIF}
3048: { Service routines }
3049: procedure NotImplemented;
3050: procedure ResourceNotFound(ResID: PChar);
```

```
3051: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3052: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3053: function PaletteColor(Color: TColor): Longint;
3054: function WidthOf(R: TRect): Integer;
3055: function HeightOf(R: TRect): Integer;
3056: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3057: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3058: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3059: procedure Delay(MSecs: Longint);
3060: procedure CenterControl(Control: TControl);
3061: Function PaletteEntries( Palette : HPALETTE) : Integer
3062: Function WindowClassName( Wnd : HWND) : string
3063: Function ScreenWorkArea : TRect
3064: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3065: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3066: Procedure ActivateWindow( Wnd : HWND)
3067: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3068: Procedure CenterWindow( Wnd : HWND)
3069: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3070: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3071: Function DialogsToPixelsX( Dlgs : Word) : Word
3072: Function DialogsToPixelsY( Dlgs : Word) : Word
3073: Function PixelsToDialogsX( Pixs : Word) : Word
3074: Function PixelsToDialogsY( Pixs : Word) : Word
3075: {$IFDEF WIN32}
3076: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3077: function MakeVariant(const Values: array of Variant): Variant;
3078: {$ENDIF}
3079: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3080: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3081: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3082: {$IFDEF CBUILDER}
3083: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3084: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3085: {$ELSE}
3086: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3087: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3088: {$ENDIF CBUILDER}
3089: function IsForegroundTask: Boolean;
3090: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3091: function GetAveCharSize(Canvas: TCanvas): TPoint;
3092: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3093: procedure FreeUnusedOle;
3094: procedure Beep;
3095: function GetWindowsVersionJ: string;
3096: function LoadDLL(const LibName: string): THandle;
3097: function RegisterServer(const ModuleName: string): Boolean;
3098: {$IFNDEF WIN32}
3099: function IsLibrary: Boolean;
3100: {$ENDIF}
3101: { Gradient filling routine }
3102: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3103: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
      TFillDirection; Colors: Byte);
3104: { String routines }
3105: function GetEnvVar(const VarName: string): string;
3106: function AnsiUpperFirstChar(const S: string): string;
3107: function StringToPChar(var S: string): PChar;
3108: function StrPAlloc(const S: string): PChar;
3109: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3110: function DropT(const S: string): string;
3111: { Memory routines }
3112: function AllocMemo(Size: Longint): Pointer;
3113: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3114: procedure FreeMemo(var fpBlock: Pointer);
3115: function GetMemoSize(fpBlock: Pointer): Longint;
3116: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3117: {$IFNDEF COMPILER5_UP}
3118: procedure FreeAndNil(var Obj);
3119: {$ENDIF}
3120: // from PNGLoader
3121: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3122: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3123: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3124: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3125: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
      TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3126:   Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3127:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3128:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3129:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3130:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3131:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3132:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3133:  Procedure SetImeName( Name : TImeName)
3134:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3135:  Function Imm32GetContext( hWnd : HWND) : HIMC
3136:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3137:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
```

```
3138:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3139:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3140: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3141:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3142:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3143:  Function Imm32IsIME( hKl : longword) : Boolean
3144:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3145:  Procedure DragDone( Drop : Boolean)
3146:
3147:
3148: //****************************************added  from jvjvclutils
3149: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3150: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3151: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3152: function IsPositiveResult(Value: TModalResult): Boolean;
3153: function IsNegativeResult(Value: TModalResult): Boolean;
3154: function IsAbortResult(const Value: TModalResult): Boolean;
3155: function StripAllFromResult(const Value: TModalResult): TModalResult;
3156: // returns either BrightColor or DarkColor depending on the luminance of AColor
3157: // This function gives the same result (AFAIK) as the function used in Windows to
3158: // calculate the desktop icon text color based on the desktop background color
3159: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3160: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3161:
3162: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3163:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3164:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3165:   var LinkName: string; Scale: Integer = 100); overload;
3166: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3167:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3168:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3169:   var LinkName: string; Scale: Integer = 100); overload;
3170: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3171:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3172: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3173:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3174:   Scale: Integer = 100): string;
3175: function HTMLPlainText(const Text: string): string;
3176: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3177:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3178: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3179:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3180: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3181: function HTMLPrepareText(const Text: string): string;
3182:
3183: ****************************************** uPSI_JvAppUtils;
3184: Function GetDefaultSection( Component : TComponent) : string
3185: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3186: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3187: Function GetDefaultIniName : string
3188:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3189: Function GetDefaultIniRegKey : string
3190: Function FindForm( FormClass : TFormClass) : TForm
3191: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3192: Function ShowDialog( FormClass : TFormClass) : Boolean
3193:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3194: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3195: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3196: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3197: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3198: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3199: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3200: Function StrToIniStr( const Str : string) : string
3201: Function IniStrToStr( const Str : string) : string
3202: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3203: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3204: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3205: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3206: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3207: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
3208: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3209: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3210: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3211: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3212: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3213: Procedure AppTaskbarIcons( AppOnly : Boolean)
3214: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3215: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3216: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3217: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3218: ****************************************** uPSI_JvDBUtils;
3219: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3220: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3221: Procedure RefreshQuery( Query : TDataSet)
3222: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3223: Function DataSetSectionName( DataSet : TDataSet) : string
3224: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string)
3225: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3226: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
      Options: TLocateOptions) : Boolean
```

```
3227: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3228: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3229: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3230: Function ConfirmDelete : Boolean
3231: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3232: Procedure CheckRequiredField( Field : TField)
3233: Procedure CheckRequiredFields( const Fields : array of TField)
3234: Function DateToSQL( Value : TDateTime) : string
3235: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3236: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3237: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
      HighEmpty:Double;Inclusive:Bool):string
3238: Function StrMaskSQL( const Value : string) : string
3239: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3240: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3241: Procedure _DBError( const Msg : string)
3242:  Const('TrueExpr','String '0=0
3243:  Const('sdfStandard16','String '''"''mm''/''dd''/''yyyy''"''
3244:  Const('sdfStandard32','String ''''''dd/mm/yyyy''''''
3245:  Const('sdfOracle','String '"TO_DATE('''"dd/mm/yyyy"''', ''DD/MM/YYYY'')"
3246:  Const('sdfInterbase','String '"CAST('''"mm"/"dd"/"yyyy"'' AS DATE)"
3247:  Const('sdfMSSQL','String '"CONVERT(datetime, '''"mm"/"dd"/"yyyy"''', 103)"
3248:  AddTypeS('Largeint', 'Longint
3249:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3250:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3251:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3252:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3253:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3254: (*----------------------------------------------------------------------------*)
3255: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3256: begin
3257:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3258:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3259:  Function JIniReadString( const FileName, Section, Line : string) : string
3260:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3261:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3262:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3263:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3264:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3265: end;
3266:
3267: (* === compile-time registration functions === *)
3268: (*----------------------------------------------------------------------------*)
3269: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3270: begin
3271:  'UnixTimeStart','LongInt'( 25569);
3272:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3273:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3274:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3275:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3276:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3277:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3278:  Function DayOfDate( const DateTime : TDateTime) : Integer
3279:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3280:  Function YearOfDate( const DateTime : TDateTime) : Integer
3281:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3282:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3283:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3284:  Function HourOfTime( const DateTime : TDateTime) : Integer
3285:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3286:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3287:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3288:  Function IsISOLongYear( const Year : Word) : Boolean;
3289:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3290:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3291:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3292:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3293:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3294:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3295:  Function JIsLeapYear( const Year : Integer) : Boolean;
3296:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3297:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3298:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3299:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3300:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3301:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3302:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3303:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3304:  Function HoursToMSecs( Hours : Integer) : Integer
3305:  Function MinutesToMSecs( Minutes : Integer) : Integer
3306:  Function SecondsToMSecs( Seconds : Integer) : Integer
3307:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3308:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3309:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3310:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3311:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3312:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3313:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3314:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
```

```
3315:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3316:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3317:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3318:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3319:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3320:  Function DosDateTimeToStr( DateTime : Integer) : string
3321:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3322:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3323:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3324:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3325:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3326:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3327:  Function FileTimeToStr( const FileTime : TFileTime) : string
3328:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3329:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3330:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3331:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3332:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3333:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3334:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3335:   TJclUnixTime32', 'Longword
3336:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3337:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3338:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3339:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3340:  Function JNullStamp : TTimeStamp
3341:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3342:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3343:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3344:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3345:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3346:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3347:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3348:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3349:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3350:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3351:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3352:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3353:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3354:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3355:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3356:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3357:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3358:   FindClass('TOBJECT'),'EJclDateTimeError
3359: end;
3360:
3361: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3362: begin
3363:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3364:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3365:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3366:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3367:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3368:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3369:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3370:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3371:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3372:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3373:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3374:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3375:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3376:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3377:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3378:  Function AbortShutDown : Boolean;
3379:  Function AbortShutDown1( const MachineName : string) : Boolean;
3380:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3381:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3382:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3383:   FindClass('TOBJECT'),'EJclCreateProcessError
3384:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3385:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
     Environment:PChar);
3386:   // with Add(EJclCreateProcessError) do
3387:  end;
3388:
3389:
3390: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3391: begin
3392:   //'AnsiSigns','Set').SetSet(['-', '+']);
3393:  'C1_UPPER','LongWord( $0001);
3394:  'C1_LOWER','LongWord( $0002);
3395:  'C1_DIGIT','LongWord').SetUInt( $0004);
3396:  'C1_SPACE','LongWord').SetUInt( $0008);
3397:  'C1_PUNCT','LongWord').SetUInt( $0010);
3398:  'C1_CNTRL','LongWord').SetUInt( $0020);
3399:  'C1_BLANK','LongWord').SetUInt( $0040);
3400:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3401:  'C1_ALPHA','LongWord').SetUInt( $0100);
3402:   AnsiChar', 'Char
```

```
3403:  Function StrIsAlpha( const S : AnsiString) : Boolean
3404:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3405:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3406:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3407:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3408:  Function StrIsDigit( const S : AnsiString) : Boolean
3409:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3410:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3411:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3412:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3413:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3414:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3415:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3416:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3417:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3418:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3419:  Function StrEscapedToString( const S : AnsiString) : AnsiString
3420:  Function JStrLower( const S : AnsiString) : AnsiString
3421:  Procedure StrLowerInPlace( var S : AnsiString)
3422:  ///Procedure StrLowerBuff( S : PAnsiChar)
3423:  Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3424:  Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3425:  Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3426:  Function StrProper( const S : AnsiString) : AnsiString
3427:  //Procedure StrProperBuff( S : PAnsiChar)
3428:  Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3429:  Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3430:  Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3431:  Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3432:  Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3433:  Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3434:  Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3435:  Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3436:  Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3437:  Function StrReverse( const S : AnsiString) : AnsiString
3438:  Procedure StrReverseInPlace( var S : AnsiString)
3439:  Function StrSingleQuote( const S : AnsiString) : AnsiString
3440:  Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
3441:  Function StrStringToEscaped( const S : AnsiString) : AnsiString
3442:  Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3443:  Function StrToHex( const Source : AnsiString) : AnsiString
3444:  Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3445:  Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3446:  Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3447:  Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3448:  Function StrTrimQuotes( const S : AnsiString) : AnsiString
3449:  Function JStrUpper( const S : AnsiString) : AnsiString
3450:  Procedure StrUpperInPlace( var S : AnsiString)
3451:  //Procedure StrUpperBuff( S : PAnsiChar)
3452:  Function StrOemToAnsi( const S : AnsiString) : AnsiString
3453:  Function StrAnsiToOem( const S : AnsiString) : AnsiString
3454:  Procedure StrAddRef( var S : AnsiString)
3455:  Function StrAllocSize( const S : AnsiString) : Longint
3456:  Procedure StrDecRef( var S : AnsiString)
3457:  //Function StrLen( S : PAnsiChar) : Integer
3458:  Function StrLength( const S : AnsiString) : Longint
3459:  Function StrRefCount( const S : AnsiString) : Longint
3460:  Procedure StrResetLength( var S : AnsiString)
3461:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3462:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3463:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3464:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3465:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3466:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3467:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3468:  Function StrFillChar(const C: Char; Count: Integer): string)');
3469:  Function IntFillChar(const I: Integer; Count: Integer): string)');
3470:  Function ByteFillChar(const B: Byte; Count: Integer): string)');
3471:  Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;');
3472:  Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3473:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3474:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3475:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3476:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
3477:  Function StrIPos( const SubStr, S : AnsiString) : Integer
3478:  Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3479:  Function StrLastPos( const SubStr, S : AnsiString) : Integer
3480:  Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3481:  Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3482:  Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3483:  Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3484:  Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3485:  Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3486:  //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3487:  //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3488:  Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3489:  Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3490:  Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3491:  Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
```

```
3492:  Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3493:  Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3494:  Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3495:  Function CharIsAlpha( const C : AnsiChar) : Boolean
3496:  Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3497:  Function CharIsBlank( const C : AnsiChar) : Boolean
3498:  Function CharIsControl( const C : AnsiChar) : Boolean
3499:  Function CharIsDelete( const C : AnsiChar) : Boolean
3500:  Function CharIsDigit( const C : AnsiChar) : Boolean
3501:  Function CharIsLower( const C : AnsiChar) : Boolean
3502:  Function CharIsNumberChar( const C : AnsiChar) : Boolean
3503:  Function CharIsPrintable( const C : AnsiChar) : Boolean
3504:  Function CharIsPunctuation( const C : AnsiChar) : Boolean
3505:  Function CharIsReturn( const C : AnsiChar) : Boolean
3506:  Function CharIsSpace( const C : AnsiChar) : Boolean
3507:  Function CharIsUpper( const C : AnsiChar) : Boolean
3508:  Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3509:  Function CharType( const C : AnsiChar) : Word
3510:  Function CharHex( const C : AnsiChar) : Byte
3511:  Function CharLower( const C : AnsiChar) : AnsiChar
3512:  Function CharUpper( const C : AnsiChar) : AnsiChar
3513:  Function CharToggleCase( const C : AnsiChar) : AnsiChar
3514:  Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3515:  Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3516:  Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer) : Integer
3517:  Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar) : Integer
3518:  Procedure StrIToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3519:  Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3520:  Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3521:  Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)
3522:  Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)
3523:  Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)
3524:  Function AddStringToStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3525:  Function BooleanToStr( B : Boolean) : AnsiString
3526:  Function FileToString( const FileName : AnsiString) : AnsiString
3527:  Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean)
3528:  Function StrToken( var S : AnsiString; Separator : AnsiChar) : AnsiString
3529:  Procedure StrTokens( const S : AnsiString; const List : TStrings)
3530:  Procedure StrTokenToStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings)
3531:  //Function StrWord( var S : PAnsiChar; out Word : AnsiString) : Boolean
3532:  Function StrToFloatSafe( const S : AnsiString) : Float
3533:  Function StrToIntSafe( const S : AnsiString) : Integer
3534:  Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer);
3535:  Function ArrayOf( List : TStrings) : TDynStringArray;
3536:   EJclStringError', 'EJclError
3537:  function IsClass(Address: TObject): Boolean;
3538:  function IsObject(Address: TObject): Boolean;
3539:  // Console Utilities
3540:  //function ReadKey: Char;
3541:  function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3542:  function JclGUIDToString(const GUID: TGUID): string;
3543:  function JclStringToGUID(const S: string): TGUID;
3544:
3545: end;
3546:
3547:
3548: ****************************************** uPSI_JvDBUtil;
3549: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const
     Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3550: Function GetQueryResult( const DatabaseName, SQL : string) : Variant
3551: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant;
     const AResultName : string) : Variant
3552: //Function StrFieldDesc( Field : FLDDesc) : string
3553: Function Var2Type( V : Variant; const VarType : Integer) : Variant
3554: Procedure CopyRecord( DataSet : TDataSet)
3555: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string;
     MasterField : Word; ModOp, DelOp : RINTQual)
3556: Procedure AddMasterPassword( Table : TTable; pswd : string)
3557: Procedure PackTable( Table : TTable)
3558: Procedure PackEncryptedTable( Table : TTable; pswd : string)
3559: Function EncodeQuotes( const S : string) : string
3560: Function Cmp( const S1, S2 : string) : Boolean
3561: Function SubStr( const S : string; const Index : Integer; const Separator : string) : string
3562: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string) : string
3563: Function ReplaceString( S : string; const OldPattern, NewPattern : string) : string
3564: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer)
3565: **********************************************uPSI_JvJvBDEUtils;***************
3566: //JvBDEUtils
3567: Function CreateDbLocate : TJvLocateObject
3568:  //Function CheckOpen( Status : DBIResult) : Boolean
3569: Procedure FetchAllRecords( DataSet : TBDEDataSet)
3570: Function TransActive( Database : TDatabase) : Boolean
3571: Function AsyncQrySupported( Database : TDatabase) : Boolean
3572: Function GetQuoteChar( Database : TDatabase) : string
3573: Procedure ExecuteQuery( const DbName, QueryText : string)
3574: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string)
3575: Function FieldLogicMap( FldType : TFieldType) : Integer
3576: Function FieldSubtypeMap( FldType : TFieldType) : Integer Value : string; Buffer : Pointer)
3577: Function GetAliasPath( const AliasName : string) : string
```

```
3578: Function IsDirectory( const DatabaseName : string) : Boolean
3579: Function GetBdeDirectory : string
3580: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean
3581: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3582: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3583: Function DataSetRecNo( DataSet : TDataSet) : Longint
3584: Function DataSetRecordCount( DataSet : TDataSet) : Longint
3585: Function DataSetPositionStr( DataSet : TDataSet) : string
3586: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)
3587: Function CurrentRecordDeleted( DataSet : TBDEDataSet) : Boolean
3588: Function IsFilterApplicable( DataSet : TDataSet) : Boolean
3589: Function IsBookmarkStable( DataSet : TBDEDataSet) : Boolean
3590: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3591: Procedure RestoreIndex( Table : TTable)
3592: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3593: Procedure PackTable( Table : TTable)
3594: Procedure ReindexTable( Table : TTable)
3595: Procedure BdeFlushBuffers
3596: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer
3597: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3598: Procedure DbNotSupported
3599: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3600: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3601: Procedure
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3602: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3603: ******************************************uPSI_JvDateUtil;
3604: function CurrentYear: Word;
3605: function IsLeapYear(AYear: Integer): Boolean;
3606: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3607: function FirstDayOfPrevMonth: TDateTime;
3608: function LastDayOfPrevMonth: TDateTime;
3609: function FirstDayOfNextMonth: TDateTime;
3610: function ExtractDay(ADate: TDateTime): Word;
3611: function ExtractMonth(ADate: TDateTime): Word;
3612: function ExtractYear(ADate: TDateTime): Word;
3613: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3614: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3615: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3616: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3617: function ValidDate(ADate: TDateTime): Boolean;
3618: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3619: function MonthsBetween(Date1, Date2: TDateTime): Double;
3620: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3621: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3622: function DaysBetween(Date1, Date2: TDateTime): Longint;
3623: { The same as previous but if Date2 < Date1 result = 0 }
3624: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3625: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3626: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3627: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3628: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3629: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3630: { String to date conversions }
3631: function GetDateOrder(const DateFormat: string): TDateOrder;
3632: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3633: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3634: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3635: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3636: function DefDateFormat(FourDigitYear: Boolean): string;
3637: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3638: ------------------------------------------------------------------------------
3639: ******************************** JvUtils;*****************************
3640: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3641: function GetWordOnPos(const S: string; const P: Integer): string;
3642: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3643: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3644: { SubStr returns substring from string, S, separated with Separator string}
3645: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3646: { SubStrEnd same to previous function but Index numerated from the end of string }
3647: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3648: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3649: function SubWord(P: PChar; var P2: PChar): string;
3650: { NumberByWord returns the text representation of
3651:   the number, N, in normal russian language. Was typed from Monitor magazine }
3652: function NumberByWord(const N: Longint): string;
3653: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3654:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3655: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3656: { GetXYByPos is same to previous function, but returns X position in line too}
3657: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3658: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3659: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3660: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3661: function ConcatSep(const S, S2, Separator: string): string;
3662: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3663: function ConcatLeftSep(const S, S2, Separator: string): string;
```

```
3664: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3665: function MinimizeString(const S: string; const MaxLen: Integer): string;
3666: { Next 4 function for russian chars transliterating.
3667:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3668: procedure Dos2Win(var S: string);
3669: procedure Win2Dos(var S: string);
3670: function Dos2WinRes(const S: string): string;
3671: function Win2DosRes(const S: string): string;
3672: function Win2Koi(const S: string): string;
3673: { Spaces returns string consists on N space chars }
3674: function Spaces(const N: Integer): string;
3675: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3676: function AddSpaces(const S: string; const N: Integer): string;
3677: { function LastDate for russian users only } { returns date relative to current date: '' }
3678: function LastDate(const Dat: TDateTime): string;
3679: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3680: function CurrencyToStr(const Cur: currency): string;
3681: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3682: function Cmp(const S1, S2: string): Boolean;
3683: { StringCat add S2 string to S1 and returns this string }
3684: function StringCat(var S1: string; S2: string): string;
3685: { HasChar returns True, if Char, Ch, contains in string, S }
3686: function HasChar(const Ch: Char; const S: string): Boolean;
3687: function HasAnyChar(const Chars: string; const S: string): Boolean;
3688: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3689: function CountOfChar(const Ch: Char; const S: string): Integer;
3690: function DefStr(const S: string; Default: string): string;
3691: {**** files routines}
3692: { GetWinDir returns Windows folder name }
3693: function GetWinDir: TFileName;
3694: function GetSysDir: String;
3695: { GetTempDir returns Windows temporary folder name }
3696: function GetTempDir: string;
3697: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3698: function GenTempFileName(FileName: string): string;
3699: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3700: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3701: { ClearDir clears folder Dir }
3702: function ClearDir(const Dir: string): Boolean;
3703: { DeleteDir clears and than delete folder Dir }
3704: function DeleteDir(const Dir: string): Boolean;
3705: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3706: function FileEquMask(FileName, Mask: TFileName): Boolean;
3707: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3708:   Masks must be separated with comma (';') }
3709: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3710: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3711: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3712: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3713: { FileGetInfo fills SearchRec record for specified file attributes}
3714: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3715: { HasSubFolder returns True, if folder APath contains other folders }
3716: function HasSubFolder(APath: TFileName): Boolean;
3717: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3718: function IsEmptyFolder(APath: TFileName): Boolean;
3719: { AddSlash add slash Char to Dir parameter, if needed }
3720: procedure AddSlash(var Dir: TFileName);
3721: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3722: function AddSlash2(const Dir: TFileName): string;
3723: { AddPath returns FileName with Path, if FileName not contain any path }
3724: function AddPath(const FileName, Path: TFileName): TFileName;
3725: function AddPaths(const PathList, Path: string): string;
3726: function ParentPath(const Path: TFileName): TFileName;
3727: function FindInPath(const FileName, PathList: string): TFileName;
3728: function FindInPaths(const fileName,paths: String): String;
3729: {$IFNDEF BCB1}
3730: { BrowseForFolder displays Browse For Folder dialog }
3731: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3732: {$ENDIF BCB1}
3733: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3734: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3735: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3736: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3737:
3738: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3739: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3740: { HasParam returns True, if program running with specified parameter, Param }
3741: function HasParam(const Param: string): Boolean;
3742: function HasSwitch(const Param: string): Boolean;
3743: function Switch(const Param: string): string;
3744: { ExePath returns ExtractFilePath(ParamStr(0)) }
3745: function ExePath: TFileName;
3746: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3747: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3748: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3749: {**** Graphic routines }
3750: { TTFontSelected returns True, if True Type font is selected in specified device context }
```

```
3751: function TTFontSelected(const DC: HDC): Boolean;
3752: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3753: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3754: {**** Windows routines }
3755: { SetWindowTop put window to top without recreating window }
3756: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3757: {**** other routines }
3758: { KeyPressed returns True, if Key VK is now pressed }
3759: function KeyPressed(VK: Integer): Boolean;
3760: procedure SwapInt(var Int1, Int2: Integer);
3761: function IntPower(Base, Exponent: Integer): Integer;
3762: function ChangeTopException(E: TObject): TObject;
3763: function StrToBool(const S: string): Boolean;
3764: {$IFNDEF COMPILER3_UP}
3765: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3766:   Length of MaxLen bytes. The compare operation is controlled by the
3767:   current Windows locale. The return value is the same as for CompareStr. }
3768: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3769: function AnsiStrIComp(S1, S2: PChar): Integer;
3770: {$ENDIF}
3771: function Var2Type(V: Variant; const VarType: Integer): Variant;
3772: function VarToInt(V: Variant): Integer;
3773: function VarToFloat(V: Variant): Double;
3774: { following functions are not documented because they are don't work properly , so don't use them }
3775: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3776: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3777: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3778: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3779: function GetParameter: string;
3780: function GetLongFileName(FileName: string): string;
3781: {* from  FileCtrl}
3782: function DirectoryExists(const Name: string): Boolean;
3783: procedure ForceDirectories(Dir: string);
3784: {# from  FileCtrl}
3785: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3786: function GetComputerID: string;
3787: function GetComputerName: string;
3788: {**** string routines }
3789: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
     same Index.Also see RAUtilsW.ReplaceSokr1 function }
3790: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3791: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3792:   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 }
3793: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3794: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3795: function CountOfLines(const S: string): Integer;
3796: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3797: procedure DeleteEmptyLines(Ss: TStrings);
3798: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3799:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3800: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3801: {**** files routines - }
3802: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3803:   Resource can be compressed using MS Compress program}
3804: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3805: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3806: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3807: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3808: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3809: { IniReadSection read section, Section, from ini-file,
3810:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3811:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3812: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3813: { LoadTextFile load text file, FileName, into string }
3814: function LoadTextFile(const FileName: TFileName): string;
3815: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3816: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3817: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3818: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3819: {$IFDEF COMPILER3_UP}
3820: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3821: function TargetFileName(const FileName: TFileName): TFileName;
3822: { return filename ShortCut linked to }
3823: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3824: {$ENDIF COMPILER3_UP}
3825: {**** Graphic routines - }
3826: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3827: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3828: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3829: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3830: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3831: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3832: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3833: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3834: { Cinema draws some visual effect }
3835: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3836: { Roughed fills rect with special 3D pattern }
3837: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
```

```
3838: { 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 }
3839: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3840: { TextWidth calculate text with for writing using standard desktop font }
3841: function TextWidth(AStr: string): Integer;
3842: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3843: function DefineCursor(Identifer: PChar): TCursor;
3844: {**** other routines - }
3845: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3846: function FindFormByClass(FormClass: TFormClass): TForm;
3847: function FindFormByClassName(FormClassName: string): TForm;
3848: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3849:   having Tag property value, equaled to Tag parameter }
3850: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3851: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3852: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3853: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3854: function RBTag(Parent: TWinControl): Integer;
3855: { AppMinimized returns True, if Application is minimized }
3856: function AppMinimized: Boolean;
3857: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3858:   if Caption parameter = '', it replaced with Application.Title }
3859: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3860: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3861:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3862: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3863:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3864: { Delay stop program execution to MSec msec }
3865: procedure Delay(MSec: Longword);
3866: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3867: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3868: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3869: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3870: function PanelBorder(Panel: TCustomPanel): Integer;
3871: function Pixels(Control: TControl; APixels: Integer): Integer;
3872: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3873: procedure Error(const Msg: string);
3874: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3875:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3876:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3877: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3878:   const HideSelColor: Boolean): string;
3879: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3880:   const HideSelColor: Boolean): Integer;
3881: function ItemHtPlain(const Text: string): string;
3882: { ClearList - clears list of TObject }
3883: procedure ClearList(List: TList);
3884: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3885: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3886: { RTTI support }
3887: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3888: function GetPropStr(Obj: TObject; const PropName: string): string;
3889: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3890: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3891: procedure PrepareIniSection(SS: TStrings);
3892: { following functions are not documented because they are don't work properly, so don't use them }
3893: {$IFDEF COMPILER2}
3894: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3895: {$ENDIF}
3896:
3897: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3898: begin
3899:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3900:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3901:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
3902:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3903:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3904:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3905:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3906:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3907: end;
3908:
3909: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3910: begin
3911:  Const('MaxInitStrNum','LongInt'( 9);
3912: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
      : array of AnsiString; MaxSplit : Integer) : Integer
3913: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
      string; MaxSplit : Integer) : Integer
3914: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
      QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3915: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3916: Function JvStrStrip( S : string) : string
3917: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3918: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3919: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3920: Function StrEatWhiteSpace( const S : string) : string
3921: Function HexToAscii( const S : AnsiString) : AnsiString
```

```
3922: Function AsciiToHex( const S : AnsiString) : AnsiString
3923: Function StripQuotes( const S1 : AnsiString) : AnsiString
3924: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3925: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3926: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3927: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3928: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3929: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
3930: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
3931: Function JvValidIdentifier( S1 : String) : Boolean
3932: Function JvEndChar( X : AnsiChar) : Boolean
3933: Procedure JvGetToken( S1, S2 : PAnsiChar)
3934: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
3935: Function IsKeyword( S1 : PAnsiChar) : Boolean
3936: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
3937: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
3938: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
3939: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
3940: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
3941: Function GetTokenCount : Integer
3942: Procedure ResetTokenCount
3943: end;
3944:
3945: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3946: begin
3947:   SIRegister_TJvQueryParamsDialog(CL);
3948:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
3949: end;
3950:
3951: ******************************** JvStrUtil /  JvStrUtils;****************************
3952: function FindNotBlankCharPos(const S: string): Integer;
3953: function AnsiChangeCase(const S: string): string;
3954: function GetWordOnPos(const S: string; const P: Integer): string;
3955: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3956: function Cmp(const S1, S2: string): Boolean;
3957: { Spaces returns string consists on N space chars }
3958: function Spaces(const N: Integer): string;
3959: { HasChar returns True, if char, Ch, contains in string, S }
3960: function HasChar(const Ch: Char; const S: string): Boolean;
3961: function HasAnyChar(const Chars: string; const S: string): Boolean;
3962: { SubStr returns substring from string, S, separated with Separator string}
3963: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3964: { SubStrEnd same to previous function but Index numerated from the end of string }
3965: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3966: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3967: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3968: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3969: { GetXYByPos is same to previous function, but returns X position in line too}
3970: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3971: { AddSlash returns string with added slash char to Dir parameter, if needed }
3972: function AddSlash2(const Dir: TFileName): string;
3973: { AddPath returns FileName with Path, if FileName not contain any path }
3974: function AddPath(const FileName, Path: TFileName): TFileName;
3975: { ExePath returns ExtractFilePath(ParamStr(0)) }
3976: function ExePath: TFileName;
3977: function LoadTextFile(const FileName: TFileName): string;
3978: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3979: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3980: function ConcatSep(const S, S2, Separator: string): string;
3981: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3982: function FileEquMask(FileName, Mask: TFileName): Boolean;
3983: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3984:   Masks must be separated with comma (';') }
3985: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3986: function StringEndsWith(const Str, SubStr: string): Boolean;
3987: function ExtractFilePath2(const FileName: string): string;
3988: function StrToOem(const AnsiStr: string): string;
3989: { StrToOem translates a string from the Windows character set into the OEM character set. }
3990: function OemToAnsiStr(const OemStr: string): string;
3991: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
3992: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
3993: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
3994: function ReplaceStr(const S, Srch, Replace: string): string;
3995: { Returns string with every occurrence of Srch string replaced with Replace string. }
3996: function DelSpace(const S: string): string;
3997: { DelSpace return a string with all white spaces removed. }
3998: function DelChars(const S: string; Chr: Char): string;
3999: { DelChars return a string with all Chr characters removed. }
4000: function DelBSpace(const S: string): string;
4001: { DelBSpace trims leading spaces from the given string. }
4002: function DelESpace(const S: string): string;
4003: { DelESpace trims trailing spaces from the given string. }
4004: function DelRSpace(const S: string): string;
4005: { DelRSpace trims leading and trailing spaces from the given string. }
4006: function DelSpace1(const S: string): string;
4007: { DelSpace1 return a string with all non-single white spaces removed. }
4008: function Tab2Space(const S: string; Numb: Byte): string;
4009: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4010: function NPos(const C: string; S: string; N: Integer): Integer;
```

```
4011: { NPos searches for a N-th position of substring C in a given string. }
4012: function MakeStr(C: Char; N: Integer): string;
4013: function MS(C: Char; N: Integer): string;
4014: { MakeStr return a string of length N filled with character C. }
4015: function AddChar(C: Char; const S: string; N: Integer): string;
4016: { AddChar return a string left-padded to length N with characters C. }
4017: function AddCharR(C: Char; const S: string; N: Integer): string;
4018: { AddCharR return a string right-padded to length N with characters C. }
4019: function LeftStr(const S: string; N: Integer): string;
4020: { LeftStr return a string right-padded to length N with blanks. }
4021: function RightStr(const S: string; N: Integer): string;
4022: { RightStr return a string left-padded to length N with blanks. }
4023: function CenterStr(const S: string; Len: Integer): string;
4024: { CenterStr centers the characters in the string based upon the Len specified. }
4025: function CompStr(const S1, S2: string): Integer;
4026: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4027: function CompText(const S1, S2: string): Integer;
4028: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4029: function Copy2Symb(const S: string; Symb: Char): string;
4030: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4031: function Copy2SymbDel(var S: string; Symb: Char): string;
4032: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4033: function Copy2Space(const S: string): string;
4034: { Copy2Symb returns a substring of a string S from begining to first white space. }
4035: function Copy2SpaceDel(var S: string): string;
4036: { Copy2SpaceDel returns a substring of a string S from begining to first
4037:   white space and removes this substring from S. }
4038: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4039: { Returns string, with the first letter of each word in uppercase,
4040:   all other letters in lowercase. Words are delimited by WordDelims. }
4041: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4042: { WordCount given a set of word delimiters, returns number of words in S. }
4043: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4044: { Given a set of word delimiters, returns start position of N'th word in S. }
4045: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4046: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4047: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4048: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4049:   delimiters, return the N'th word in S. }
4050: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4051: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4052: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4053: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4054: function QuotedString(const S: string; Quote: Char): string;
4055: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4056: function ExtractQuotedString(const S: string; Quote: Char): string;
4057: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4058:    and reduces pairs of Quote characters within the quoted string to a single character. }
4059: function FindPart(const HelpWilds, InputStr: string): Integer;
4060: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4061: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4062: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4063: function XorString(const Key, Src: ShortString): ShortString;
4064: function XorEncode(const Key, Source: string): string;
4065: function XorDecode(const Key, Source: string): string;
4066: { ** Command line routines ** }
4067: {$IFNDEF COMPILER4_UP}
4068: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4069: {$ENDIF}
4070: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4071: { ** Numeric string handling routines ** }
4072: function Numb2USA(const S: string): string;
4073: { Numb2USA converts numeric string S to USA-format. }
4074: function Dec2Hex(N: Longint; A: Byte): string;
4075: function D2H(N: Longint; A: Byte): string;
4076: { Dec2Hex converts the given value to a hexadecimal string representation
4077:   with the minimum number of digits (A) specified. }
4078: function Hex2Dec(const S: string): Longint;
4079: function H2D(const S: string): Longint;
4080: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4081: function Dec2Numb(N: Longint; A, B: Byte): string;
4082: { Dec2Numb converts the given value to a string representation with the
4083:   base equal to B and with the minimum number of digits (A) specified. }
4084: function Numb2Dec(S: string; B: Byte): Longint;
4085: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4086: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4087: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4088: function IntToRoman(Value: Longint): string;
4089: { IntToRoman converts the given value to a roman numeric string representation. }
4090: function RomanToInt(const S: string): Longint;
4091: { RomanToInt converts the given string to an integer value. If the string
4092:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4093: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4094: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4095: ********************************* JvFileUtil;*********************************
4096: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4097: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4098: procedure MoveFile(const FileName, DestName: TFileName);
```

```
4099: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4100: {$IFDEF COMPILER4_UP}
4101: function GetFileSize(const FileName: string): Int64;
4102: {$ELSE}
4103: function GetFileSize(const FileName: string): Longint;
4104: {$ENDIF}
4105: function FileDateTime(const FileName: string): TDateTime;
4106: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4107: function DeleteFiles(const FileMask: string): Boolean;
4108: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4109: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4110: function NormalDir(const DirName: string): string;
4111: function RemoveBackSlash(const DirName: string): string;
4112: function ValidFileName(const FileName: string): Boolean;
4113: function DirExists(Name: string): Boolean;
4114: procedure ForceDirectories(Dir: string);
4115: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4116:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4117: {$IFDEF COMPILER4_UP}
4118: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4119: {$ENDIF}
4120: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4121: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4122: {$IFDEF COMPILER4_UP}
4123: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4124: {$ENDIF}
4125: function GetTempDir: string;
4126: function GetWindowsDir: string;
4127: function GetSystemDir: string;
4128: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4129: {$IFDEF WIN32}
4130: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4131: function ShortToLongFileName(const ShortName: string): string;
4132: function ShortToLongPath(const ShortName: string): string;
4133: function LongToShortFileName(const LongName: string): string;
4134: function LongToShortPath(const LongName: string): string;
4135: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4136: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4137: {$ENDIF WIN32}
4138: {$IFNDEF COMPILER3_UP}
4139: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4140: {$ENDIF}
4141: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4142: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4143: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4144: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4145:
4146: //**************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4147:  Procedure VariantClear( var V : Variant)
4148:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4149:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4150:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4151:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4152:  Procedure VariantSub( const src : Variant; var dst : Variant)
4153:  Procedure VariantMul( const src : Variant; var dst : Variant)
4154:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4155:  Procedure VariantMod( const src : Variant; var dst : Variant)
4156:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4157:  Procedure VariantOr( const src : Variant; var dst : Variant)
4158:  Procedure VariantXor( const src : Variant; var dst : Variant)
4159:  Procedure VariantShl( const src : Variant; var dst : Variant)
4160:  Procedure VariantShr( const src : Variant; var dst : Variant)
4161:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4162:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4163:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4164:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4165:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4166:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4167:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4168:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4169:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4170:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4171:  Function VariantNot( const V1 : Variant) : Variant
4172:  Function VariantNeg( const V1 : Variant) : Variant
4173:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4174:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4175:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4176:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4177:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4178:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4179:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4180:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4181:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4182:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4183: end;
4184:
4185: ************************************unit uPSI_JvgUtils;************************************
4186: function IsEven(I: Integer): Boolean;
4187: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
```

```
4188: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4189: procedure SwapInt(var I1, I2: Integer);
4190: function Spaces(Count: Integer): string;
4191: function DupStr(const Str: string; Count: Integer): string;
4192: function DupChar(C: Char; Count: Integer): string;
4193: procedure Msg(const AMsg: string);
4194: function RectW(R: TRect): Integer;
4195: function RectH(R: TRect): Integer;
4196: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4197: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4198: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4199: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4200:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4201: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4202: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4203:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4204: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4205: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4206:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4207: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4208: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4209: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4210:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4211:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4212:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4213: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4214:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4215:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4216:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4217: procedure BringParentWindowToTop(Wnd: TWinControl);
4218: function GetParentForm(Control: TControl): TForm;
4219: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4220: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4221: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4222: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4223: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4224: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4225: function CalcMathString(AExpression: string): Single;
4226: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4227: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4228: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4229: procedure TypeStringOnKeyboard(const S: string);
4230: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4231: procedure DrawTextExtAligned(Canvas:TCanvas;const
      Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4232: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4233: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4234: function ComponentToString(Component: TComponent): string;
4235: procedure StringToComponent(Component: TComponent; const Value: string);
4236: function PlayWaveResource(const ResName: string): Boolean;
4237: function UserName: string;
4238: function ComputerName: string;
4239: function CreateIniFileName: string;
4240: function ExpandString(const Str: string; Len: Integer): string;
4241: function Transliterate(const Str: string; RusToLat: Boolean): string;
4242: function IsSmallFonts: Boolean;
4243: function SystemColorDepth: Integer;
4244: function GetFileTypeJ(const FileName: string): TglFileType;
4245: Function GetFileType( hFile : THandle) : DWORD');
4246: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4247: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4248:
4249: { ************************************Utility routines of unit classes}
4250: function LineStart(Buffer, BufPos: PChar): PChar
4251: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4252:   'Strings: TStrings): Integer
4253:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4254:  Procedure RegisterClass( AClass : TPersistentClass)
4255:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4256:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4257:  Procedure UnRegisterClass( AClass : TPersistentClass)
4258:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4259:  Procedure UnRegisterModuleClasses( Module : HMODULE)
4260:  Function FindGlobalComponent( const Name : string) : TComponent
4261:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4262:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4263:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4264:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4265:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4266:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4267:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4268:  Procedure GlobalFixupReferences
4269:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4270:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4271:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4272:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4273:  Procedure RemoveFixups( Instance : TPersistent)
4274:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
```

```
4275:  Procedure BeginGlobalLoading
4276:  Procedure NotifyGlobalLoading
4277:  Procedure EndGlobalLoading
4278:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4279:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4280:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4281:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4282:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
4283: // Function AllocateHWnd( Method : TWndMethod) : HWND
4284:  Procedure DeallocateHWnd( Wnd : HWND)
4285:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4286: ************************************unit uPSI_SqlTimSt and DB;*********************************
4287: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4288: Function VarSQLTimeStampCreate3: Variant;
4289: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4290: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4291: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4292: Function VarSQLTimeStamp : TVarType
4293: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4294: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4295: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4296: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4297: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4298: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4299: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4300: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4301: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4302: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4303: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4304: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4305: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4306:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4307: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4308: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4309: Procedure DisposeMem( var Buffer, Size : Integer)
4310: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4311: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4312: Function VarTypeToDataType( VarType : Integer) : TFieldType
4313: ********************************************unit JvStrings;*************************************
4314: {template functions}
4315: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4316: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4317: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4318: function RemoveMasterBlocks(const SourceStr: string): string;
4319: function RemoveFields(const SourceStr: string): string;
4320: {http functions}
4321: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4322: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4323: {set functions}
4324: procedure SplitSet(AText: string; AList: TStringList);
4325: function JoinSet(AList: TStringList): string;
4326: function FirstOfSet(const AText: string): string;
4327: function LastOfSet(const AText: string): string;
4328: function CountOfSet(const AText: string): Integer;
4329: function SetRotateRight(const AText: string): string;
4330: function SetRotateLeft(const AText: string): string;
4331: function SetPick(const AText: string; AIndex: Integer): string;
4332: function SetSort(const AText: string): string;
4333: function SetUnion(const Set1, Set2: string): string;
4334: function SetIntersect(const Set1, Set2: string): string;
4335: function SetExclude(const Set1, Set2: string): string;
4336: {replace any <,> etc by &lt; &gt;}
4337: function XMLSafe(const AText: string): string;
4338: {simple hash, Result can be used in Encrypt}
4339: function Hash(const AText: string): Integer;
4340: { Base64 encode and decode a string }
4341: function B64Encode(const S: AnsiString): AnsiString;
4342: function B64Decode(const S: AnsiString): AnsiString;
4343: {Basic encryption from a Borland Example}
4344: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4345: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4346: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4347: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4348: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4349: procedure CSVToTags(Src, Dst: TStringList);
4350: // converts a csv list to a tagged string list
4351: procedure TagsToCSV(Src, Dst: TStringList);
4352: // converts a tagged string list to a csv list
4353: // only fieldnames from the first record are scanned ib the other records
4354: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4355: {selects akey=avalue from Src and returns recordset in Dst}
4356: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4357: {filters Src for akey=avalue}
4358: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4359: {orders a tagged Src list by akey}
4360: function PosStr(const FindString, SourceString: string;
4361:   StartPos: Integer = 1): Integer;
4362: { PosStr searches the first occurrence of a substring FindString in a string
4363:   given by SourceString with case sensitivity (upper and lower case characters
```

```
4364:   are differed). This function returns the index value of the first character
4365:   of a specified substring from which it occurs in a given string starting with
4366:   StartPos character index. If a specified substring is not found Q_PosStr
4367:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4368: function PosStrLast(const FindString, SourceString: string): Integer;
4369: {finds the last occurance}
4370: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4371: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4372: { PosText searches the first occurrence of a substring FindString in a string
4373:   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
4374: function PosTextLast(const FindString, SourceString: string): Integer;
4375: {finds the last occurance}
4376: function NameValuesToXML(const AText: string): string;
4377: {$IFDEF MSWINDOWS}
4378: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4379: {$ENDIF MSWINDOWS}
4380: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4381: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4382: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4383: procedure SaveString(const AFile, AText: string);
4384: Procedure SaveStringasFile( const AFile, AText : string)
4385: function LoadStringJ(const AFile: string): string;
4386: Function LoadStringofFile( const AFile : string) : string
4387: Procedure SaveStringtoFile( const AFile, AText : string)
4388: Function LoadStringfromFile( const AFile : string) : string
4389: function HexToColor(const AText: string): TColor;
4390: function UppercaseHTMLTags(const AText: string): string;
4391: function LowercaseHTMLTags(const AText: string): string;
4392: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4393: function RelativePath(const ASrc, ADst: string): string;
4394: function GetToken(var Start: Integer; const SourceText: string): string;
4395: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4396: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4397: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4398: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4399: // parses the beginning of an attribute: space + alpha character
4400: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4401: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4402: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4403: // parses all name=value attributes to the attributes TStringList
4404: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4405: // checks if a name="value" pair exists and returns any value
4406: function GetStrValue(const AText, AName, ADefault: string): string;
4407: // retrieves string value from a line like:
4408: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4409: // returns ADefault when not found
4410: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4411: // same for a color
4412: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4413: // same for an Integer
4414: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4415: // same for a float
4416: function GetBoolValue(const AText, AName: string): Boolean;
4417: // same for Boolean but without default
4418: function GetValue(const AText, AName: string): string;
4419: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4420: procedure SetValue(var AText: string; const AName, AValue: string);
4421: // sets a string value in a line
4422: procedure DeleteValue(var AText: string; const AName: string);
4423: // deletes a AName="value" pair from AText
4424: procedure GetNames(AText: string; AList: TStringList);
4425: // get a list of names from a string with name="value" pairs
4426: function GetHTMLColor(AColor: TColor): string;
4427: // converts a color value to the HTML hex value
4428: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4429: // finds a string backward case sensitive
4430: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4431: // finds a string backward case insensitive
4432: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4433:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4434: // finds a text range, e.g. <TD>....</TD> case sensitive
4435: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4436:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4437: // finds a text range, e.g. <TD>....</td> case insensitive
4438: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4439:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4440: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4441: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4442:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4443: // finds a text range backward, e.g. <TD>....</td> case insensitive
4444: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4445:   var RangeEnd: Integer): Boolean;
4446: // finds a HTML or XML tag:  <....>
4447: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4448:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4449: // finds the innertext between opening and closing tags
4450: function Easter(NYear: Integer): TDateTime;
```

```
4451: // returns the easter date of a year.
4452: function GetWeekNumber(Today: TDateTime): string;
4453: //gets a datecode. Returns year and weeknumber in format: YYWW
4454: function ParseNumber(const S: string): Integer;
4455: // parse number returns the last position, starting from 1
4456: function ParseDate(const S: string): Integer;
4457: // parse a SQL style data string from positions 1,
4458: // starts and ends with #
4459:
4460: ****************************************unit JvJCLUtils;****************************************
4461:
4462: function VarIsInt(Value: Variant): Boolean;
4463:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4464: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4465: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4466: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4467: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4468: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4469: function GetWordOnPos(const S: string; const P: Integer): string;
4470: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4471: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4472: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4473: { GetWordOnPosEx working like GetWordOnPos function, but
4474:   also returns Word position in iBeg, iEnd variables }
4475: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4476: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4477: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4478: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4479: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4480: { GetEndPosCaret returns the caret position of the last char. For the position
4481:   after the last char of Text you must add 1 to the returned X value. }
4482: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
4483: { GetEndPosCaret returns the caret position of the last char. For the position
4484:   after the last char of Text you must add 1 to the returned X value. }
4485: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4486: function SubStrBySeparator(const S:string;const Index:Integer;const
     Separator:string;StartIndex:Int=1):string;
4487: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
     Separator:WideString;StartIndex:Int:WideString;
4488: { SubStrEnd same to previous function but Index numerated from the end of string }
4489: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4490: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4491: function SubWord(P: PChar; var P2: PChar): string;
4492: function CurrencyByWord(Value: Currency): string;
4493: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4494: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4495: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4496: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4497: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4498: { ReplaceString searches for all substrings, OldPattern,
4499:   in a string, S, and replaces them with NewPattern }
4500: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4501: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
     WideString;StartIndex:Integer=1):WideString;
4502: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4503: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4504: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4505: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4506:
4507: { Next 4 function for russian chars transliterating.
4508:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4509: procedure Dos2Win(var S: AnsiString);
4510: procedure Win2Dos(var S: AnsiString);
4511: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4512: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4513: function Win2Koi(const S: AnsiString): AnsiString;
4514: { FillString fills the string Buffer with Count Chars }
4515: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4516: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4517: { MoveString copies Count Chars from Source to Dest }
4518: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
     inline; {$ENDIF SUPPORTS_INLINE} overload;
4519: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4520:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4521: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4522: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4523: { MoveWideChar copies Count WideChars from Source to Dest }
4524: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
     SUPPORTS_INLINE}
4525: { FillNativeChar fills Buffer with Count NativeChars }
4526: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4527: { MoveWideChar copies Count WideChars from Source to Dest }
4528: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4529: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4530: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
```

```
4531: { Spaces returns string consists on N space chars }
4532: function Spaces(const N: Integer): string;
4533: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4534: function AddSpaces(const S: string; const N: Integer): string;
4535: function SpacesW(const N: Integer): WideString;
4536: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4537: { function LastDateRUS for russian users only }
4538: { returns date relative to current date: 'ääà äiý íàçàä' }
4539: function LastDateRUS(const Dat: TDateTime): string;
4540: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4541: function CurrencyToStr(const Cur: Currency): string;
4542: { HasChar returns True, if Char, Ch, contains in string, S }
4543: function HasChar(const Ch: Char; const S: string): Boolean;
4544: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4545: function HasAnyChar(const Chars: string; const S: string): Boolean;
4546: {$IFNDEF COMPILER12_UP}
4547: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4548: {$ENDIF ~COMPILER12_UP}
4549: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
     SUPPORTS_INLINE}
4550: function CountOfChar(const Ch: Char; const S: string): Integer;
4551: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4552: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4553: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4554: function StrPosW(S, SubStr: PWideChar): PWideChar;
4555: function StrLenW(S: PWideChar): Integer;
4556: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4557: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4558: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4559: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4560: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4561:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4562: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4563: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4564: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4565: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4566: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
4567: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4568: Function ScreenPixelFormat : TPixelFormat
4569: Function ScreenColorCount : Integer
4570: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4571: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4572: //  SIRegister_TJvGradient(CL);
4573:
4574: {************************************ files routines}
4575: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4576: const
4577:   {$IFDEF MSWINDOWS}
4578:   DefaultCaseSensitivity = False;
4579:   {$ENDIF MSWINDOWS}
4580:   {$IFDEF UNIX}
4581:   DefaultCaseSensitivity = True;
4582:   {$ENDIF UNIX}
4583: { GenTempFileName returns temporary file name on
4584:   drive, there FileName is placed }
4585: function GenTempFileName(FileName: string): string;
4586: { GenTempFileNameExt same to previous function, but
4587:   returning filename has given extension, FileExt }
4588: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4589: { ClearDir clears folder Dir }
4590: function ClearDir(const Dir: string): Boolean;
4591: { DeleteDir clears and than delete folder Dir }
4592: function DeleteDir(const Dir: string): Boolean;
4593: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4594: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4595: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4596:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4597: function FileEquMasks(FileName, Masks: TFileName;
4598:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4599: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4600: {$IFDEF MSWINDOWS}
4601: { LZFileExpand expand file, FileSource,
4602:   into FileDest. Given file must be compressed, using MS Compress program }
4603: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4604: {$ENDIF MSWINDOWS}
4605: { FileGetInfo fills SearchRec record for specified file attributes}
4606: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4607: { HasSubFolder returns True, if folder APath contains other folders }
4608: function HasSubFolder(APath: TFileName): Boolean;
4609: { IsEmptyFolder returns True, if there are no files or
4610:   folders in given folder, APath}
4611: function IsEmptyFolder(APath: TFileName): Boolean;
4612: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4613: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4614: { AddPath returns FileName with Path, if FileName not contain any path }
4615: function AddPath(const FileName, Path: TFileName): TFileName;
4616: function AddPaths(const PathList, Path: string): string;
```

```
4617: function ParentPath(const Path: TFileName): TFileName;
4618: function FindInPath(const FileName, PathList: string): TFileName;
4619: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4620: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4621: { HasParam returns True, if program running with specified parameter, Param }
4622: function HasParam(const Param: string): Boolean;
4623: function HasSwitch(const Param: string): Boolean;
4624: function Switch(const Param: string): string;
4625: { ExePath returns ExtractFilePath(ParamStr(0)) }
4626: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4627: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4628: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4629: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4630: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4631: {**** Graphic routines }
4632: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4633: function IsTTFontSelected(const DC: HDC): Boolean;
4634: function KeyPressed(VK: Integer): Boolean;
4635: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4636: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4637: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4638: {**** Color routines }
4639: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4640: function RGBToBGR(Value: Cardinal): Cardinal;
4641: //function ColorToPrettyName(Value: TColor): string;
4642: //function PrettyNameToColor(const Value: string): TColor;
4643: {**** other routines }
4644: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4645: function IntPower(Base, Exponent: Integer): Integer;
4646: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4647: function StrToBool(const S: string): Boolean;
4648: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4649: function VarToInt(V: Variant): Integer;
4650: function VarToFloat(V: Variant): Double;
4651: { following functions are not documented because they not work properly sometimes, so do not use them }
4652: // (rom) ReplaceStrings1, GetSubStr removed
4653: function GetLongFileName(const FileName: string): string;
4654: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4655: function GetParameter: string;
4656: function GetComputerID: string;
4657: function GetComputerName: string;
4658: {**** string routines }
4659: { ReplaceAllStrings searches for all substrings, Words,
4660:   in a string, S, and replaces them with Frases with the same Index. }
4661: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4662: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4663:   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 }
4664: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4665: { CountOfLines calculates the lines count in a string, S,
4666:   each line must be separated from another with CrLf sequence }
4667: function CountOfLines(const S: string): Integer;
4668: { DeleteLines deletes all lines from strings which in the words,  words.
4669:   The word of will be deleted from strings. }
4670: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4671: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4672:   Lines contained only spaces also deletes. }
4673: procedure DeleteEmptyLines(Ss: TStrings);
4674: { SQLAddWhere addes or modifies existing where-statement, where,
4675:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4676:   it must be started on the begining of any line }
4677: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4678: {**** files routines - }
4679: {$IFDEF MSWINDOWS}
4680: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4681:   Resource can be compressed using MS Compress program}
4682: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4683: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
     Boolean;
4684: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4685: {$ENDIF MSWINDOWS}
4686: { IniReadSection read section, Section, from ini-file,
4687:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4688:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4689: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4690: { LoadTextFile load text file, FileName, into string }
4691: function LoadTextFile(const FileName: TFileName): string;
4692: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4693: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4694: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4695: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4696: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4697: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4698: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4699: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4700: { RATextCalcHeight calculate needed height for
4701:   correct output, using RATextOut or RATextOutEx functions }
4702: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4703: { Cinema draws some visual effect }
```

```
4704: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4705: { Roughed fills rect with special 3D pattern }
4706: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4707: { 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 }
4708: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4709: { TextWidth calculate text with for writing using standard desktop font }
4710: function TextWidth(const AStr: string): Integer;
4711: { TextHeight calculate text height for writing using standard desktop font }
4712: function TextHeight(const AStr: string): Integer;
4713: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4714: procedure Error(const Msg: string);
4715: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4716:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4717: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4718: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4719:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4720: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4721:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4722: function ItemHtPlain(const Text: string): string;
4723: { ClearList - clears list of TObject }
4724: procedure ClearList(List: TList);
4725: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4726: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4727: { RTTI support }
4728: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4729: function GetPropStr(Obj: TObject; const PropName: string): string;
4730: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4731: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4732: procedure PrepareIniSection(Ss: TStrings);
4733: { following functions are not documented because they are don't work properly, so don't use them }
4734: // (rom) from JvBandWindows to make it obsolete
4735: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4736: // (rom) from JvBandUtils to make it obsolete
4737: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4738: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4739: function CreateIconFromClipboard: TIcon;
4740: { begin JvIconClipboardUtils } { Icon clipboard routines }
4741: function CF_ICON: Word;
4742: procedure AssignClipboardIcon(Icon: TIcon);
4743: { Real-size icons support routines (32-bit only) }
4744: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4745: function CreateRealSizeIcon(Icon: TIcon): HICON;
4746: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4747: {end JvIconClipboardUtils }
4748: function CreateScreenCompatibleDC: HDC;
4749: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF}
4750: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4751: { begin JvRLE } // (rom) changed API for inclusion in JCL
4752: procedure RleCompressTo(InStream, OutStream: TStream);
4753: procedure RleDecompressTo(InStream, OutStream: TStream);
4754: procedure RleCompress(Stream: TStream);
4755: procedure RleDecompress(Stream: TStream);
4756: { end JvRLE } { begin JvDateUtil }
4757: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4758: function IsLeapYear(AYear: Integer): Boolean;
4759: function DaysInAMonth(const AYear, AMonth: Word): Word;
4760: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4761: function FirstDayOfPrevMonth: TDateTime;
4762: function LastDayOfPrevMonth: TDateTime;
4763: function FirstDayOfNextMonth: TDateTime;
4764: function ExtractDay(ADate: TDateTime): Word;
4765: function ExtractMonth(ADate: TDateTime): Word;
4766: function ExtractYear(ADate: TDateTime): Word;
4767: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4768: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4769: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4770: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4771: function ValidDate(ADate: TDateTime): Boolean;
4772: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4773: function MonthsBetween(Date1, Date2: TDateTime): Double;
4774: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4775: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4776: function DaysBetween(Date1, Date2: TDateTime): Longint;
4777: { The same as previous but if Date2 < Date1 result = 0 }
4778: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4779: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4780: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4781: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4782: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4783: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4784: { String to date conversions }
4785: function GetDateOrder(const DateFormat: string): TDateOrder;
4786: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4787: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4788: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4789: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4790: //function DefDateFormat(AFourDigitYear: Boolean): string;
```

```
4791: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4792: function FormatLongDate(Value: TDateTime): string;
4793: function FormatLongDateTime(Value: TDateTime): string;
4794: { end JvDateUtil }
4795: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4796: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4797: { begin JvStrUtils } { ** Common string handling routines ** }
4798: {$IFDEF UNIX}
4799: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4800:   const ToCode, FromCode: AnsiString): Boolean;
4801: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4802: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4803: function OemStrToAnsi(const S: AnsiString): AnsiString;
4804: function AnsiStrToOem(const S: AnsiString): AnsiString;
4805: {$ENDIF UNIX}
4806: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4807: { StrToOem translates a string from the Windows character set into the OEM character set. }
4808: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4809: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4810: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4811: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4812: function ReplaceStr(const S, Srch, Replace: string): string;
4813: { Returns string with every occurrence of Srch string replaced with Replace string. }
4814: function DelSpace(const S: string): string;
4815: { DelSpace return a string with all white spaces removed. }
4816: function DelChars(const S: string; Chr: Char): string;
4817: { DelChars return a string with all Chr characters removed. }
4818: function DelBSpace(const S: string): string;
4819: { DelBSpace trims leading spaces from the given string. }
4820: function DelESpace(const S: string): string;
4821: { DelESpace trims trailing spaces from the given string. }
4822: function DelRSpace(const S: string): string;
4823: { DelRSpace trims leading and trailing spaces from the given string. }
4824: function DelSpace1(const S: string): string;
4825: { DelSpace1 return a string with all non-single white spaces removed. }
4826: function Tab2Space(const S: string; Numb: Byte): string;
4827: { Tab2Space converts any tabulation character in the given string to the
4828:   Numb spaces characters. }
4829: function NPos(const C: string; S: string; N: Integer): Integer;
4830: { NPos searches for a N-th position of substring C in a given string. }
4831: function MakeStr(C: Char; N: Integer): string; overload;
4832: {$IFNDEF COMPILER12_UP}
4833: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4834: {$ENDIF !COMPILER12_UP}
4835: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4836: { MakeStr return a string of length N filled with character C. }
4837: function AddChar(C: Char; const S: string; N: Integer): string;
4838: { AddChar return a string left-padded to length N with characters C. }
4839: function AddCharR(C: Char; const S: string; N: Integer): string;
4840: { AddCharR return a string right-padded to length N with characters C. }
4841: function LeftStr(const S: string; N: Integer): string;
4842: { LeftStr return a string right-padded to length N with blanks. }
4843: function RightStr(const S: string; N: Integer): string;
4844: { RightStr return a string left-padded to length N with blanks. }
4845: function CenterStr(const S: string; Len: Integer): string;
4846: { CenterStr centers the characters in the string based upon the Len specified. }
4847: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4848: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4849:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4850: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4851: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4852: function Copy2Symb(const S: string; Symb: Char): string;
4853: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4854: function Copy2SymbDel(var S: string; Symb: Char): string;
4855: { Copy2SymbDel returns a substring of a string S from begining to first
4856:   character Symb and removes this substring from S. }
4857: function Copy2Space(const S: string): string;
4858: { Copy2Symb returns a substring of a string S from begining to first white space. }
4859: function Copy2SpaceDel(var S: string): string;
4860: { Copy2SpaceDel returns a substring of a string S from begining to first
4861:   white space and removes this substring from S. }
4862: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4863: { Returns string, with the first letter of each word in uppercase,
4864:   all other letters in lowercase. Words are delimited by WordDelims. }
4865: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4866: { WordCount given a set of word delimiters, returns number of words in S. }
4867: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4868: { Given a set of word delimiters, returns start position of N'th word in S. }
4869: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4870: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4871: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4872: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4873:   delimiters, return the N'th word in S. }
4874: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4875: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4876:   that started from position Pos. }
4877: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4878: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4879: function QuotedString(const S: string; Quote: Char): string;
```

```
4880: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4881: function ExtractQuotedString(const S: string; Quote: Char): string;
4882: { ExtractQuotedString removes the Quote characters from the beginning and
4883:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4884: function FindPart(const HelpWilds, InputStr: string): Integer;
4885: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4886: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4887: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4888: function XorString(const Key, Src: ShortString): ShortString;
4889: function XorEncode(const Key, Source: string): string;
4890: function XorDecode(const Key, Source: string): string;
4891: { ** Command line routines ** }
4892: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4893: { ** Numeric string handling routines ** }
4894: function Numb2USA(const S: string): string;
4895: { Numb2USA converts numeric string S to USA-format. }
4896: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4897: { Dec2Hex converts the given value to a hexadecimal string representation
4898:   with the minimum number of digits (A) specified. }
4899: function Hex2Dec(const S: string): Longint;
4900: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4901: function Dec2Numb(N: Int64; A, B: Byte): string;
4902: { Dec2Numb converts the given value to a string representation with the
4903:   base equal to B and with the minimum number of digits (A) specified. }
4904: function Numb2Dec(S: string; B: Byte): Int64;
4905: { Numb2Dec converts the given B-based numeric string to the corresponding
4906:   integer value. }
4907: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4908: { IntToBin converts the given value to a binary string representation
4909:   with the minimum number of digits specified. }
4910: function IntToRoman(Value: Longint): string;
4911: { IntToRoman converts the given value to a roman numeric string representation. }
4912: function RomanToInt(const S: string): Longint;
4913: { RomanToInt converts the given string to an integer value. If the string
4914:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4915: function FindNotBlankCharPos(const S: string): Integer;
4916: function FindNotBlankCharPosW(const S: WideString): Integer;
4917: function AnsiChangeCase(const S: string): string;
4918: function WideChangeCase(const S: string): string;
4919: function StartsText(const SubStr, S: string): Boolean;
4920: function EndsText(const SubStr, S: string): Boolean;
4921: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4922: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4923: {end JvStrUtils}
4924: {$IFDEF UNIX}
4925: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4926: {$ENDIF UNIX}
4927: { begin JvFileUtil }
4928: function FileDateTime(const FileName: string): TDateTime;
4929: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4930: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4931: function NormalDir(const DirName: string): string;
4932: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4933: function ValidFileName(const FileName: string): Boolean;
4934: {$IFDEF MSWINDOWS}
4935: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4936: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4937: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4938: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4939: {$ENDIF MSWINDOWS}
4940: function GetWindowsDir: string;
4941: function GetSystemDir: string;
4942: function ShortToLongFileName(const ShortName: string): string;
4943: function LongToShortFileName(const LongName: string): string;
4944: function ShortToLongPath(const ShortName: string): string;
4945: function LongToShortPath(const LongName: string): string;
4946: {$IFDEF MSWINDOWS}
4947: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4948: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4949: {$ENDIF MSWINDOWS}
4950: { end JvFileUtil }
4951: // Works like PtInRect but includes all edges in comparision
4952: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4953: // Works like PtInRect but excludes all edges from comparision
4954: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4955: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4956: function IsFourDigitYear: Boolean;
4957: { moved from JvJVCLUTils }
4958: //Open an object with the shell (url or something like that)
4959: function OpenObject(const Value: string): Boolean; overload;
4960: function OpenObject(Value: PChar): Boolean; overload;
4961: {$IFDEF MSWINDOWS}
4962: //Raise the last Exception
4963: procedure RaiseLastWin32; overload;
4964: procedure RaiseLastWin32(const Text: string); overload;
4965: //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
4966: function GetFileVersion(const AFileName: string): Cardinal;
```

```
4967: {$EXTERNALSYM GetFileVersion}
4968: //Get version of Shell.dll
4969: function GetShellVersion: Cardinal;
4970: {$EXTERNALSYM GetShellVersion}
4971: // CD functions  on HW
4972: procedure OpenCdDrive;
4973: procedure CloseCdDrive;
4974: // returns True if Drive is accessible
4975: function DiskInDrive(Drive: Char): Boolean;
4976: {$ENDIF MSWINDOWS}
4977: //Same as linux function ;)
4978: procedure PError(const Text: string);
4979: // execute a program without waiting
4980: procedure Exec(const FileName, Parameters, Directory: string);
4981: // execute a program and wait for it to finish
4982: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
4983: // returns True if this is the first instance of the program that is running
4984: function FirstInstance(const ATitle: string): Boolean;
4985: // restores a window based on it's classname and Caption. Either can be left empty
4986: // to widen the search
4987: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
4988: // manipulate the traybar and start button
4989: procedure HideTraybar;
4990: procedure ShowTraybar;
4991: procedure ShowStartButton(Visible: Boolean = True);
4992: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
4993: procedure MonitorOn;
4994: procedure MonitorOff;
4995: procedure LowPower;
4996: // send a key to the window named AppName
4997: function SendKey(const AppName: string; Key: Char): Boolean;
4998: {$IFDEF MSWINDOWS}
4999: // returns a list of all win currently visible, the Objects property is filled with their window handle
5000: procedure GetVisibleWindows(List: TStrings);
5001: Function GetVisibleWindowsF( List : TStrings):TStrings');
5002: // associates an extension to a specific program
5003: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5004: procedure AddToRecentDocs(const FileName: string);
5005: function GetRecentDocs: TStringList;
5006: {$ENDIF MSWINDOWS}
5007: function CharIsMoney(const Ch: Char): Boolean;
5008: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5009: function IntToExtended(I: Integer): Extended;
5010: { GetChangedText works out the new text given the current cursor pos & the key pressed
5011:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5012: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5013: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5014: //function StrIsInteger(const S: string): Boolean;
5015: function StrIsFloatMoney(const Ps: string): Boolean;
5016: function StrIsDateTime(const Ps: string): Boolean;
5017: function PreformatDateString(Ps: string): string;
5018: function BooleanToInteger(const B: Boolean): Integer;
5019: function StringToBoolean(const Ps: string): Boolean;
5020: function SafeStrToDateTime(const Ps: string): TDateTime;
5021: function SafeStrToDate(const Ps: string): TDateTime;
5022: function SafeStrToTime(const Ps: string): TDateTime;
5023: function StrDelete(const psSub, psMain: string): string;
5024:   { returns the fractional value of pcValue}
5025: function TimeOnly(pcValue: TDateTime): TTime;
5026: { returns the integral value of pcValue }
5027: function DateOnly(pcValue: TDateTime): TDate;
5028: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5029: const { TDateTime value used to signify Null value}
5030:   NullEquivalentDate: TDateTime = 0.0;
5031: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5032: // Replacement for Win32Check to avoid platform specific warnings in D6
5033: function OSCheck(RetVal: Boolean): Boolean;
5034: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5035:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5036:   not be forced to use FileCtrl unnecessarily }
5037: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5038: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5039: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5040: function MinimizeString(const S: string; const MaxLen: Integer): string;
5041: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
     SW_SHOWDEFAULT);
5042: { 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.}
5043: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5044: {$ENDIF MSWINDOWS}
5045: procedure ResourceNotFound(ResID: PChar);
5046: function EmptyRect: TRect;
5047: function RectWidth(R: TRect): Integer;
5048: function RectHeight(R: TRect): Integer;
5049: function CompareRect(const R1, R2: TRect): Boolean;
5050: procedure RectNormalize(var R: TRect);
5051: function RectIsSquare(const R: TRect): Boolean;
5052: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
```

```
5053: //If AMaxSize = -1 ,then auto calc Square's max size
5054: {$IFDEF MSWINDOWS}
5055: procedure FreeUnusedOle;
5056: function GetWindowsVersion: string;
5057: function LoadDLL(const LibName: string): THandle;
5058: function RegisterServer(const ModuleName: string): Boolean;
5059: function UnregisterServer(const ModuleName: string): Boolean;
5060: {$ENDIF MSWINDOWS}
5061: { String routines }
5062: function GetEnvVar(const VarName: string): string;
5063: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5064: function StringToPChar(var S: string): PChar;
5065: function StrPAlloc(const S: string): PChar;
5066: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5067: function DropT(const S: string): string;
5068: { Memory routines }
5069: function AllocMemo(Size: Longint): Pointer;
5070: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5071: procedure FreeMemo(var fpBlock: Pointer);
5072: function GetMemoSize(fpBlock: Pointer): Longint;
5073: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5074: { Manipulate huge pointers routines }
5075: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5076: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5077: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5078: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5079: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5080: function WindowClassName(Wnd: THandle): string;
5081: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5082: procedure ActivateWindow(Wnd: THandle);
5083: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5084: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5085: { SetWindowTop put window to top without recreating window }
5086: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5087: procedure CenterWindow(Wnd: THandle);
5088: function MakeVariant(const Values: array of Variant): Variant;
5089: { Convert dialog units to pixels and backwards }
5090: {$IFDEF MSWINDOWS}
5091: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5092: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5093: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5094: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5095: {$ENDIF MSWINDOWS}
5096: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5097: {$IFDEF BCB}
5098: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5099: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5100: {$ELSE}
5101: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5102: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5103: {$ENDIF BCB}
5104: {$IFDEF MSWINDOWS}
5105: { BrowseForFolderNative displays Browse For Folder dialog }
5106: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5107: {$ENDIF MSWINDOWS}
5108: procedure AntiAlias(Clip: TBitmap);
5109: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5110: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5111:   ABitmap: TBitmap; const SourceRect: TRect);
5112: function IsTrueType(const FontName: string): Boolean;
5113: // Removes all non-numeric characters from AValue and returns the resulting string
5114: function TextToValText(const AValue: string): string;
5115: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5116: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5117: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5118: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5119: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5120:
5121: ****************************************************unit uPSI_JvTFUtils;
5122:  Function JExtractYear( ADate : TDateTime) : Word
5123:  Function JExtractMonth( ADate : TDateTime) : Word
5124:  Function JExtractDay( ADate : TDateTime) : Word
5125:  Function ExtractHours( ATime : TDateTime) : Word
5126:  Function ExtractMins( ATime : TDateTime) : Word
5127:  Function ExtractSecs( ATime : TDateTime) : Word
5128:  Function ExtractMSecs( ATime : TDateTime) : Word
5129:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5130:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5131:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5132:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5133:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5134:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5135:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5136:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5137:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5138:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5139:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5140:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
```

```
5141:  Procedure EnsureMonth( Month : Word)
5142:  Procedure EnsureDOW( DOW : Word)
5143:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5144:  Function Lesser( N1, N2 : Integer) : Integer
5145:  Function Greater( N1, N2 : Integer) : Integer
5146:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5147:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5148:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5149:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5150:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5151:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5152:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
      AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5153:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
      HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5154:  Function JRectWidth( ARect : TRect) : Integer
5155:  Function JRectHeight( ARect : TRect) : Integer
5156:  Function JEmptyRect : TRect
5157:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5158:
5159: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5160: begin
5161:  Procedure HideTaskBarButton( hWindow : HWND)
5162:  Function msLoadStr( ID : Integer) : String
5163:  Function msFormat( fmt : String; params : array of const) : String
5164:  Function msFileExists( const FileName : String) : Boolean
5165:  Function msIntToStr( Int : Int64) : String
5166:  Function msStrPas( const Str : PChar) : String
5167:  Function msRenameFile( const OldName, NewName : String) : Boolean
5168:  Function CutFileName( s : String) : String
5169:  Function GetVersionInfo( var VersionString : String) : DWORD
5170:  Function FormatTime( t : Cardinal) : String
5171:  Function msCreateDir( const Dir : string) : Boolean
5172:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5173:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5174:  Function msStrLen( Str : PChar) : Integer
5175:  Function msDirectoryExists( const Directory : String) : Boolean
5176:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5177:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5178:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5179:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5180:  Function GetTextFromFile( Filename : String) : string
5181:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5182:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5183:  Function msStrToInt( s : String) : Integer
5184:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
5185: end;
5186:
5187: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5188: begin
5189:   //TDynFloatArray', 'array of Extended
5190:   TDynLWordArray', 'array of LongWord
5191:   TDynLIntArray', 'array of LongInt
5192:   TDynFloatMatrix', 'array of TDynFloatArray
5193:   TDynLWordMatrix', 'array of TDynLWordArray
5194:   TDynLIntMatrix', 'array of TDynLIntArray
5195:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5196:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5197:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5198:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5199:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5200:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5201:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5202:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5203:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5204:  Function MNorm( const X : TDynFloatArray) : Extended
5205:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5206:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5207:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5208:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5209:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5210:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5211:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5212:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5213:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5214:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5215:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5216:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5217:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5218: end;
5219:
5220: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5221: begin
5222:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5223:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5224:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5225:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5226:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5227:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
```

```
5228:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5229:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5230:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5231:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5232:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5233:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5234:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5235:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5236:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5237:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5238:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5239:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5240:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5241:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5242:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5243:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5244:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5245:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5246:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5247:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5248:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5249:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5250:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5251:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
5252:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5253:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5254:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5255:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5256:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5257:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5258:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5259:  'ESBInvPi','Extended').setExtended( 3.18309886183790671 54e-1);
5260:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5261:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5262:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5263:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5264:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5265:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5266:  'ESBPiOn4','Extended').setExtended( 0.78539816339744830 96);
5267:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5268:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5269:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5270:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5271:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5272:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5273:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5274:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5275:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5276:   //LongWord', 'Cardinal
5277:   TBitList', 'Word
5278:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5279:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5280:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5281:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5282:  Function SameFloat( const X1, X2 : Extended) : Boolean
5283:  Function FloatIsZero( const X : Extended) : Boolean
5284:  Function FloatIsPositive( const X : Extended) : Boolean
5285:  Function FloatIsNegative( const X : Extended) : Boolean
5286:  Procedure IncLim( var B : Byte; const Limit : Byte)
5287:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5288:  Procedure IncLimW( var B : Word; const Limit : Word)
5289:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5290:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5291:  Procedure DecLim( var B : Byte; const Limit : Byte)
5292:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5293:  Procedure DecLimW( var B : Word; const Limit : Word)
5294:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5295:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5296:  Function MaxB( const B1, B2 : Byte) : Byte
5297:  Function MinB( const B1, B2 : Byte) : Byte
5298:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5299:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5300:  Function MaxW( const B1, B2 : Word) : Word
5301:  Function MinW( const B1, B2 : Word) : Word
5302:  Function esbMaxI( const B1, B2 : Integer) : Integer
5303:  Function esbMinI( const B1, B2 : Integer) : Integer
5304:  Function MaxL( const B1, B2 : LongInt) : LongInt
5305:  Function MinL( const B1, B2 : LongInt) : LongInt
5306:  Procedure SwapB( var B1, B2 : Byte)
5307:  Procedure SwapSI( var B1, B2 : ShortInt)
5308:  Procedure SwapW( var B1, B2 : Word)
5309:  Procedure SwapI( var B1, B2 : SmallInt)
5310:  Procedure SwapL( var B1, B2 : LongInt)
5311:  Procedure SwapI32( var B1, B2 : Integer)
5312:  Procedure SwapC( var B1, B2 : LongWord)
5313:  Procedure SwapInt64( var X, Y : Int64)
5314:  Function esbSign( const B : LongInt) : ShortInt
5315:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5316:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
```

```
5317:  Function Max3Word( const X1, X2, X3 : Word) : Word
5318:  Function Min3Word( const X1, X2, X3 : Word) : Word
5319:  Function MaxBArray( const B : array of Byte) : Byte
5320:  Function MaxWArray( const B : array of Word) : Word
5321:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5322:  Function MaxIArray( const B : array of Integer) : Integer
5323:  Function MaxLArray( const B : array of LongInt) : LongInt
5324:  Function MinBArray( const B : array of Byte) : Byte
5325:  Function MinWArray( const B : array of Word) : Word
5326:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5327:  Function MinIArray( const B : array of Integer) : Integer
5328:  Function MinLArray( const B : array of LongInt) : LongInt
5329:  Function SumBArray( const B : array of Byte) : Byte
5330:  Function SumBArray2( const B : array of Byte) : Word
5331:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5332:  Function SumSIArray2( const B : array of ShortInt) : Integer
5333:  Function SumWArray( const B : array of Word) : Word
5334:  Function SumWArray2( const B : array of Word) : LongInt
5335:  Function SumIArray( const B : array of Integer) : Integer
5336:  Function SumLArray( const B : array of LongInt) : LongInt
5337:  Function SumLWArray( const B : array of LongWord) : LongWord
5338:  Function ESBDigits( const X : LongWord) : Byte
5339:  Function BitsHighest( const X : LongWord) : Integer
5340:  Function ESBBitsNeeded( const X : LongWord) : Integer
5341:  Function esbGCD( const X, Y : LongWord) : LongWord
5342:  Function esbLCM( const X, Y : LongInt) : Int64
5343:  //Function esbLCM( const X, Y : LongInt) : LongInt
5344:  Function RelativePrime( const X, Y : LongWord) : Boolean
5345:  Function Get87ControlWord : TBitList
5346:  Procedure Set87ControlWord( const CWord : TBitList)
5347:  Procedure SwapExt( var X, Y : Extended)
5348:  Procedure SwapDbl( var X, Y : Double)
5349:  Procedure SwapSing( var X, Y : Single)
5350:  Function esbSgn( const X : Extended) : ShortInt
5351:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5352:  Function ExtMod( const X, Y : Extended) : Extended
5353:  Function ExtRem( const X, Y : Extended) : Extended
5354:  Function CompMOD( const X, Y : Comp) : Comp
5355:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5356:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5357:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5358:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5359:  Function MaxExt( const X, Y : Extended) : Extended
5360:  Function MinExt( const X, Y : Extended) : Extended
5361:  Function MaxEArray( const B : array of Extended) : Extended
5362:  Function MinEArray( const B : array of Extended) : Extended
5363:  Function MaxSArray( const B : array of Single) : Single
5364:  Function MinSArray( const B : array of Single) : Single
5365:  Function MaxCArray( const B : array of Comp) : Comp
5366:  Function MinCArray( const B : array of Comp) : Comp
5367:  Function SumSArray( const B : array of Single) : Single
5368:  Function SumEArray( const B : array of Extended) : Extended
5369:  Function SumSqEArray( const B : array of Extended) : Extended
5370:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5371:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5372:  Function SumCArray( const B : array of Comp) : Comp
5373:  Function FactorialX( A : LongWord) : Extended
5374:  Function PermutationX( N, R : LongWord) : Extended
5375:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5376:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5377:  Function esbGeometricMean( const X : array of Extended) : Extended
5378:  Function esbHarmonicMean( const X : array of Extended) : Extended
5379:  Function ESBMean( const X : array of Extended) : Extended
5380:  Function esbSampleVariance( const X : array of Extended) : Extended
5381:  Function esbPopulationVariance( const X : array of Extended) : Extended
5382:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5383:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5384:  Function GetMedian( const SortedX : array of Extended) : Extended
5385:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5386:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5387:  Function ESBMagnitude( const X : Extended) : Integer
5388:  Function ESBTan( Angle : Extended) : Extended
5389:  Function ESBCot( Angle : Extended) : Extended
5390:  Function ESBCosec( const Angle : Extended) : Extended
5391:  Function ESBSec( const Angle : Extended) : Extended
5392:  Function ESBArcTan( X, Y : Extended) : Extended
5393:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5394:  Function ESBArcCos( const X : Extended) : Extended
5395:  Function ESBArcSin( const X : Extended) : Extended
5396:  Function ESBArcSec( const X : Extended) : Extended
5397:  Function ESBArcCosec( const X : Extended) : Extended
5398:  Function ESBLog10( const X : Extended) : Extended
5399:  Function ESBLog2( const X : Extended) : Extended
5400:  Function ESBLogBase( const X, Base : Extended) : Extended
5401:  Function Pow2( const X : Extended) : Extended
5402:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5403:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5404:  Function XtoY( const X, Y : Extended) : Extended
5405:  Function esbTenToY( const Y : Extended) : Extended
```

```
5406:  Function esbTwoToY( const Y : Extended) : Extended
5407:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5408:  Function esbISqrt( const I : LongWord) : Longword
5409:  Function ILog2( const I : LongWord) : LongWord
5410:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5411:  Function ESBArCosh( X : Extended) : Extended
5412:  Function ESBArSinh( X : Extended) : Extended
5413:  Function ESBArTanh( X : Extended) : Extended
5414:  Function ESBCosh( X : Extended) : Extended
5415:  Function ESBSinh( X : Extended) : Extended
5416:  Function ESBTanh( X : Extended) : Extended
5417:  Function InverseGamma( const X : Extended) : Extended
5418:  Function esbGamma( const X : Extended) : Extended
5419:  Function esbLnGamma( const X : Extended) : Extended
5420:  Function esbBeta( const X, Y : Extended) : Extended
5421:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5422: end;
5423:
5424: ******************************Integer Huge Cardinal Utils*************************
5425: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5426: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5427: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5428: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5429: Function BitCount_8( Value : byte) : integer
5430: Function BitCount_16( Value : uint16) : integer
5431: Function BitCount_32( Value : uint32) : integer
5432: Function BitCount_64( Value : uint64) : integer
5433: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5434:  Procedure ( CountPrimalityTests : integer)
5435:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5436:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5437:  Function isCoPrime( a, b : THugeCardinal) : boolean
5438:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5439:  Function hasSmallFactor( p : THugeCardinal) : boolean
5440:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer) : boolean
5441:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5442:  Const('StandardExponent','LongInt'( 65537);
5443: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5444:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5445:
5446: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5447: begin
5448:   AddTypeS('TXRTLInteger', 'array of Integer
5449:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5450:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5451:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5452:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5453:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5454:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5455:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5456:   'BitsPerByte','LongInt'( 8);
5457:   BitsPerDigit','LongInt'( 32);
5458:   SignBitMask','LongWord( $80000000);
5459:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5460:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5461:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5462:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5463:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5464:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5465:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5466:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5467:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5468:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5469:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5470:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5471:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5472:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5473:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5474:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5475:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5476:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5477:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5478:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5479:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5480:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5481:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5482:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5483:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5484:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5485:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5486:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5487:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5488:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5489:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5490:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
```

```
5491:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5492:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5493:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5494:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5495:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5496:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger)
5497:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5498:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5499:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5500:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5501:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5502:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5503:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5504:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5505:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5506:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5507:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5508:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5509:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5510:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5511:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5512:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5513:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5514:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5515:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5516:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5517:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5518:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5519:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5520:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5521:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5522:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5523:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5524:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5525:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5526:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5527:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5528:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5529:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5530:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5531:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5532:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5533: end;
5534:
5535: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5536: begin
5537:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5538:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5539:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5540:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5541:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5542:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5543:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5544:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5545:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5546:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5547:  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)
5548: end;
5549:
5550:
5551: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5552: begin
5553:  Function StrDec( S : String) : String
5554:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5555:  Function ReadNumFromEdit( Edit : TEdit) : Float
5556:  Procedure WriteNumToFile( var F : Text; X : Float)
5557: end;
5558:
5559: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5560: begin
5561:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5562:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5563:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5564:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5565:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5566:  Procedure TeX_SetGraphTitle( Title : String)
5567:  Procedure TeX_SetOxTitle( Title : String)
5568:  Procedure TeX_SetOyTitle( Title : String)
5569:  Procedure TeX_PlotOxAxis
5570:  Procedure TeX_PlotOyAxis
```

```
5571:  Procedure TeX_PlotGrid( Grid : TGrid)
5572:  Procedure TeX_WriteGraphTitle
5573:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5574:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5575:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5576:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5577:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5578:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5579:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5580:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5581:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5582:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5583:  Function Xcm( X : Float) : Float
5584:  Function Ycm( Y : Float) : Float
5585: end;
5586:
5587: *---------------------------------------------------------------------------*)
5588: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5589: begin
5590:   TConstArray', 'array of TVarRec
5591:  Function CopyVarRec( const Item : TVarRec) : TVarRec
5592:  Function CreateConstArray( const Elements : array of const) : TConstArray
5593:  Procedure FinalizeVarRec( var Item : TVarRec)
5594:  Procedure FinalizeConstArray( var Arr : TConstArray)
5595: end;
5596:
5597: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5598: begin
5599:  Function HexBS( B : Byte) : ShortString
5600:  Function HexWS( W : Word) : ShortString
5601:  Function HexLS( L : LongInt) : ShortString
5602:  Function HexPtrS( P : Pointer) : ShortString
5603:  Function BinaryBS( B : Byte) : ShortString
5604:  Function BinaryWS( W : Word) : ShortString
5605:  Function BinaryLS( L : LongInt) : ShortString
5606:  Function OctalBS( B : Byte) : ShortString
5607:  Function OctalWS( W : Word) : ShortString
5608:  Function OctalLS( L : LongInt) : ShortString
5609:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5610:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5611:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5612:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5613:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5614:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5615:  Function Long2StrS( L : LongInt) : ShortString
5616:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5617:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5618:  Function ValPrepS( const S : ShortString) : ShortString
5619:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5620:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5621:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5622:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5623:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5624:  Function TrimLeadS( const S : ShortString) : ShortString
5625:  Function TrimTrailS( const S : ShortString) : ShortString
5626:  Function TrimS( const S : ShortString) : ShortString
5627:  Function TrimSpacesS( const S : ShortString) : ShortString
5628:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5629:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5630:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5631:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5632:  Function ScrambleS( const S, Key : ShortString) : ShortString
5633:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5634:  Function FilterS( const S, Filters : ShortString) : ShortString
5635:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5636:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5637:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5638:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5639:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5640:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5641:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5642:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5643:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
5644:  Function CompStringS( const S1, S2 : ShortString) : Integer
5645:  Function CompUCStringS( const S1, S2 : ShortString) : Integer
5646:  Function SoundexS( const S : ShortString) : ShortString
5647:  Function MakeLetterSetS( const S : ShortString) : Longint
5648:  Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5649:  Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5650:  Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
      Pos:Cardinal):Bool;
5651:  Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5652:  Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5653:  Function JustFilenameS( const PathName : ShortString) : ShortString
5654:  Function JustNameS( const PathName : ShortString) : ShortString
5655:  Function JustExtensionS( const Name : ShortString) : ShortString
5656:  Function JustPathnameS( const PathName : ShortString) : ShortString
```

```
5657:  Function AddBackSlashS( const DirName : ShortString) : ShortString
5658:  Function CleanPathNameS( const PathName : ShortString) : ShortString
5659:  Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5660:  Function CommaizeS( L : LongInt) : ShortString
5661:  Function CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5662:  Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
      DecPt:Char):ShortString;
5663:  Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5664:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5665:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5666:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5667:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5668:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5669:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5670:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5671:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5672:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5673:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5674:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5675:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5676:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5677:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
      SubString:ShortString):Bool;
5678:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
      SubStr:ShortString):Bool;
5679:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5680:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5681:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5682:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5683:  Function ExtractTokensS(const S,
      Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5684:  Function IsChAlphaS( C : Char) : Boolean
5685:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5686:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5687:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5688:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5689:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5690:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5691:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5692:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5693:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5694:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5695:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
      Cardinal):ShortString;
5696:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
      Replacements:Cardinal):ShortString;
5697:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5698:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
      Replacements:Cardinal):ShortString
5699:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
      Replacements:Cardinal):ShortString
5700:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5701:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5702:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5703:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5704: end;
5705:
5706:
5707: ********unit uPSI_StUtils; from Systools4*****************************************************
5708: Function SignL( L : LongInt) : Integer
5709: Function SignF( F : Extended) : Integer
5710: Function MinWord( A, B : Word) : Word
5711: Function MidWord( W1, W2, W3 : Word) : Word
5712: Function MaxWord( A, B : Word) : Word
5713: Function MinLong( A, B : LongInt) : LongInt
5714: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5715: Function MaxLong( A, B : LongInt) : LongInt
5716: Function MinFloat( F1, F2 : Extended) : Extended
5717: Function MidFloat( F1, F2, F3 : Extended) : Extended
5718: Function MaxFloat( F1, F2 : Extended) : Extended
5719: Function MakeInteger16( H, L : Byte) : SmallInt
5720: Function MakeWordS( H, L : Byte) : Word
5721: Function SwapNibble( B : Byte) : Byte
5722: Function SwapWord( L : LongInt) : LongInt
5723: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5724: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5725: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5726: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5727: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5728: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5729: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5730: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5731: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5732: Procedure ExchangeBytes( var I, J : Byte)
5733: Procedure ExchangeWords( var I, J : Word)
5734: Procedure ExchangeLongInts( var I, J : LongInt)
```

```
5735: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5736: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5737: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5738: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5739: //*****************uPSI_StFIN;************************************************************
5740: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5741: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
      Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5742: Function BondDuration( Settlement,Maturity:TStDate;Rate,
      Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5743: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
      Extended
5744: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5745: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5746: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5747: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5748: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5749: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5750: Function DollarToDecimalText( DecDollar : Extended) : string
5751: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5752: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5753: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5754: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5755: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5756: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5757: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5758: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5759: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5760: Function IsCardValid( const S : string) : Boolean
5761: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5762: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5763: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5764: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5765: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5766: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5767: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5768: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5769: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5770: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5771: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5772: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5773: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5774: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5775: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5776: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5777: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5778: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5779: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5780: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5781:
5782: //*****************************************unit uPSI_StAstroP;
5783: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5784: //*****unit unit uPSI_StStat; Statistic Package of SysTools***************************************
5785: Function AveDev( const Data : array of Double) : Double
5786: Function AveDev16( const Data, NData : Integer) : Double
5787: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5788: Function Correlation( const Data1, Data2 : array of Double) : Double
5789: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5790: Function Covariance( const Data1, Data2 : array of Double) : Double
5791: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5792: Function DevSq( const Data : array of Double) : Double
5793: Function DevSq16( const Data, NData : Integer) : Double
5794: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5795:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5796: Function GeometricMeanS( const Data : array of Double) : Double
5797: Function GeometricMean16( const Data, NData : Integer) : Double
5798: Function HarmonicMeanS( const Data : array of Double) : Double
5799: Function HarmonicMean16( const Data, NData : Integer) : Double
5800: Function Largest( const Data : array of Double; K : Integer) : Double
5801: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5802: Function MedianS( const Data : array of Double) : Double
5803: Function Median16( const Data, NData : Integer) : Double
5804: Function Mode( const Data : array of Double) : Double
5805: Function Mode16( const Data, NData : Integer) : Double
5806: Function Percentile( const Data : array of Double; K : Double) : Double
5807: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5808: Function PercentRank( const Data : array of Double; X : Double) : Double
5809: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5810: Function Permutations( Number, NumberChosen : Integer) : Extended
5811: Function Combinations( Number, NumberChosen : Integer) : Extended
```

```
5812: Function FactorialS( N : Integer) : Extended
5813: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5814: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5815: Function Smallest( const Data : array of Double; K : Integer) : Double
5816: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5817: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5818: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5819:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5820:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5821: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
     LF:TStLinEst;ErrorStats:Bool;
5822: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
     LF:TStLinEst;ErrorStats:Bool;
5823: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5824: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5825: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5826: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5827: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5828: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5829: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5830: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5831: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5832: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5833: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5834: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5835: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5836: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5837: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5838: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5839: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5840: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5841: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5842: Function NormSDist( Z : Single) : Single
5843: Function NormSInv( Probability : Single) : Single
5844: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5845: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5846: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5847: Function Erfc( X : Single) : Single
5848: Function GammaLn( X : Single) : Single
5849: Function LargestSort( const Data : array of Double; K : Integer) : Double
5850: Function SmallestSort( const Data : array of double; K : Integer) : Double
5851:
5852: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5853:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5854:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5855:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5856:  Function DefaultMergeName( MergeNum : Integer) : string
5857:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5858:
5859: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5860: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5861:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5862:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5863:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5864:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5865:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5866:  Function LunarPhase( UT : TStDateTimeRec) : Double
5867:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5868:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5869:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5870:  Function FullMoon( D : TStDate) : TStLunarRecord
5871:  Function LastQuarter( D : TStDate) : TStLunarRecord
5872:  Function NewMoon( D : TStDate) : TStLunarRecord
5873:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5874:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5875:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5876:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5877:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5878:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5879:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5880:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5881:  Function SiderealTime( UT : TStDateTimeRec) : Double
5882:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5883:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5884:  Function SEaster( Y, Epoch : Integer) : TStDate
5885:  Function DateTimeToAJD( D : TDateTime) : Double
5886:  Function HoursMin( RA : Double) : ShortString
5887:  Function DegsMin( DC : Double) : ShortString
5888:  Function AJDToDateTime( D : Double) : TDateTime
5889:
5890: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5891:  Function CurrentDate : TStDate
5892:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5893:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5894:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5895:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5896:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5897:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5898:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
```

```
5899:  Function WeekOfYear( Julian : TStDate) : Byte
5900:  Function AstJulianDate( Julian : TStDate) : Double
5901:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5902:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5903:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5904:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5905:  Function StIsLeapYear( Year : Integer) : Boolean
5906:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5907:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5908:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5909:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5910:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5911:  Function CurrentTime : TStTime
5912:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5913:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5914:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5915:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5916:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5917:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5918:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5919:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5920:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5921:  Function StDateToDateTime( D : TStDate) : TDateTime
5922:  Function StTimeToDateTime( T : TStTime) : TDateTime
5923:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5924:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5925:
5926: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5927: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5928:  Function MonthToString( const Month : Integer) : string
5929:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5930:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5931:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5932:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5933:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5934:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5935:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5936:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5937:  Function TimeStringToStTime( const Picture, S : string) : TStTime
5938:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5939:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5940:  Function DateStringIsBlank( const Picture, S : string) : Boolean
5941:  Function InternationalDate( ForceCentury : Boolean) : string
5942:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5943:  Function InternationalTime( ShowSeconds : Boolean) : string
5944:  Procedure ResetInternationalInfo
5945:
5946: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5947:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5948:  Function AnsiUpperCaseShort32( const S : string) : string
5949:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
5950:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
5951:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
5952:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5953:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5954:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5955:  Function Upcase( C : AnsiChar) : AnsiChar
5956:  Function LoCase( C : AnsiChar) : AnsiChar
5957:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
5958:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
5959:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
5960:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5961:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5962:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
5963:  Procedure RaiseContainerError( Code : longint)
5964:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
5965:  Function ProductOverflow( A, B : LongInt) : Boolean
5966:  Function StNewStr( S : string) : PShortString
5967:  Procedure StDisposeStr( PS : PShortString)
5968:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
5969:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
5970:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
5971:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
5972:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
5973:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
5974:
5975: procedure SIRegister_usvd(CL: TPSPascalCompiler);
5976: begin
5977:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
5978:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
5979:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
5980:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
5981:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
5982: end;
5983:
5984: //**********unit unit ; StMath Package of SysTools****************************************
5985: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
5986: Function PowerS( Base, Exponent : Extended) : Extended
5987: Function StInvCos( X : Double) : Double
```

```
5988: Function StInvSin( Y : Double) : Double
5989: Function StInvTan2( X, Y : Double) : Double
5990: Function StTan( A : Double) : Double
5991: Procedure DumpException;  //unit StExpEng;
5992: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
5993:
5994: //**********unit unit ; StCRC Package of SysTools*****************************************
5995: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
5996: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
5997: Function Adler32OfFile( FileName : AnsiString) : LongInt
5998: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
5999: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6000: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6001: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6002: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6003: Function Crc32OfFile( FileName : AnsiString) : LongInt
6004: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6005: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6006: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6007: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6008: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6009: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6010:
6011: //**********unit unit ; StBCD Package of SysTools*****************************************
6012: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6013: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6014: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6015: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6016: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6017: Function NegBcd( const B : TbcdS) : TbcdS
6018: Function AbsBcd( const B : TbcdS) : TbcdS
6019: Function FracBcd( const B : TbcdS) : TbcdS
6020: Function IntBcd( const B : TbcdS) : TbcdS
6021: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6022: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6023: Function ValBcd( const S : string) : TbcdS
6024: Function LongBcd( L : LongInt) : TbcdS
6025: Function ExtBcd( E : Extended) : TbcdS
6026: Function ExpBcd( const B : TbcdS) : TbcdS
6027: Function LnBcd( const B : TbcdS) : TbcdS
6028: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6029: Function PowBcd( const B, E : TbcdS) : TbcdS
6030: Function SqrtBcd( const B : TbcdS) : TbcdS
6031: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6032: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6033: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6034: Function IsIntBcd( const B : TbcdS) : Boolean
6035: Function TruncBcd( const B : TbcdS) : LongInt
6036: Function BcdExt( const B : TbcdS) : Extended
6037: Function RoundBcd( const B : TbcdS) : LongInt
6038: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6039: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6040: Function FormatBcd( const Format : string; const B : TbcdS) : string
6041: Function StrGeneralBcd( const B : TbcdS) : string
6042: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6043: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6044:
6045: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6046: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6047: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6048: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6049: Function StDeEscape( const EscStr : AnsiString) : Char
6050: Function StDoEscape( Delim : Char) : AnsiString
6051: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6052: Function AnsiHashText( const S : string; Size : Integer) : Integer
6053: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6054: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6055: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6056:
6057: //**********unit unit ; StNetCon Package of SysTools*****************************************
6058:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6059:     Constructor Create( AOwner : TComponent)
6060:     Function Connect : DWord
6061:     Function Disconnect : DWord
6062:     RegisterProperty('Password', 'String', iptrw);
6063:     Property('UserName', 'String', iptrw);
6064:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6065:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6066:     Property('LocalDevice', 'String', iptrw);
6067:     Property('ServerName', 'String', iptrw);
6068:     Property('ShareName', 'String', iptrw);
6069:     Property('OnConnect', 'TNotifyEvent', iptrw);
6070:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6071:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6072:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6073:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6074:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6075:   end;
6076: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
```

```
6077: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6078: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6079: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6080: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6081: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6082: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6083: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6084: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6085: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6086: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6087: Function SuspendThread( hThread : THandle) : DWORD
6088: Function ResumeThread( hThread : THandle) : DWORD
6089: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6090: Function GetCurrentThread : THandle
6091: Procedure ExitThread( dwExitCode : DWORD)
6092: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6093: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6094: Procedure EndThread(ExitCode: Integer);
6095: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6096: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6097: Procedure FreeProcInstance( Proc : FARPROC)
6098: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6099: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6100: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6101: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6102: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6103: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6104: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6105: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6106: Function CurrentParallelJobInfo : TParallelJobInfo
6107: Function ObtainParallelJobInfo : TParallelJobInfo
6108: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6109: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6110: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6111: Function DeviceIoControl( hDevice : THandle; dwIoControlCode : DWORD; lpInBuffer : TObject; nInBufferSize
      : DWORD; lpOutBuffer: TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped :
      TOverlapped) : BOOL');
6112: Function SetFileTime( hFile : THandle; lpCreationTime, lpLastAccessTime, lpLastWriteTime : TFileTime) :
      BOOL');
6113: Function DuplicateHandle( hSourceProcessHandle, hSourceHandle, hTargetProcessHandle : THandle;
      lpTargetHandle : THandle; dwDesiredAccess : DWORD; bInheritHandle : BOOL; dwOptions : DWORD) : BOOL');
6114: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6115: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6116:
6117: ************************************************unit uPSI_JclMime;
6118:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6119:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6120:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6121:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6122:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6123:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6124:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6125:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6126:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6127:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6128:
6129: ************************************************unit uPSI_JclPrint;
6130:  Procedure DirectPrint( const Printer, Data : string)
6131:  Procedure SetPrinterPixelsPerInch
6132:  Function GetPrinterResolution : TPoint
6133:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6134:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6135:
6136:
6137: //*************************************unit uPSI_ShLwApi;************************************
6138:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6139:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6140:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6141:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6142:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6143:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6144:  Function StrDup( lpSrch : PChar) : PChar
6145:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6146:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6147:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6148:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6149: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6150:  Function StrPBrk( psz, pszSet : PChar) : PChar
6151:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6152:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6153:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6154:  Function StrSpn( psz, pszSet : PChar) : Integer
6155:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6156:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6157:  Function StrToInt( lpSrch : PChar) : Integer
6158:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
```

```
6159:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6160:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6161:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6162:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6163:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6164:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6165:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6166:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6167:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6168:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6169:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6170:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6171:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6172:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6173:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6174:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6175:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6176:  STIF_DEFAULT','LongWord( $00000000);
6177:  STIF_SUPPORT_HEX','LongWord( $00000001);
6178:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6179:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6180:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6181:  Function PathAddBackslash( pszPath : PChar) : PChar
6182:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6183:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6184:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6185:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6186:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6187:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6188:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6189:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6190:  Function PathFileExists( pszPath : PChar) : BOOL
6191:  Function PathFindExtension( pszPath : PChar) : PChar
6192:  Function PathFindFileName( pszPath : PChar) : PChar
6193:  Function PathFindNextComponent( pszPath : PChar) : PChar
6194:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6195:  Function PathGetArgs( pszPath : PChar) : PChar
6196:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6197:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6198:  Function PathGetCharType( ch : Char) : UINT
6199:  GCT_INVALID','LongWord( $0000);
6200:  GCT_LFNCHAR','LongWord( $0001);
6201:  GCT_SHORTCHAR','LongWord( $0002);
6202:  GCT_WILD','LongWord( $0004);
6203:  GCT_SEPARATOR','LongWord( $0008);
6204:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6205:  Function PathIsDirectory( pszPath : PChar) : BOOL
6206:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6207:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6208:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6209:  Function PathIsRelative( pszPath : PChar) : BOOL
6210:  Function PathIsRoot( pszPath : PChar) : BOOL
6211:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6212:  Function PathIsUNC( pszPath : PChar) : BOOL
6213:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6214:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6215:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6216:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6217:  Function PathIsURL( pszPath : PChar) : BOOL
6218:  Function PathMakePretty( pszPath : PChar) : BOOL
6219:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6220:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6221:  Procedure PathQuoteSpaces( lpsz : PChar)
6222:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6223:  Procedure PathRemoveArgs( pszPath : PChar)
6224:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6225:  Procedure PathRemoveBlanks( pszPath : PChar)
6226:  Procedure PathRemoveExtension( pszPath : PChar)
6227:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6228:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6229:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6230:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6231:  Function PathSkipRoot( pszPath : PChar) : PChar
6232:  Procedure PathStripPath( pszPath : PChar)
6233:  Function PathStripToRoot( pszPath : PChar) : BOOL
6234:  Procedure PathUnquoteSpaces( lpsz : PChar)
6235:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6236:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6237:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6238:  Procedure PathUndecorate( pszPath : PChar)
6239:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6240:  URL_SCHEME_INVALID','LongInt'( - 1);
6241:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6242:  URL_SCHEME_FTP','LongInt'( 1);
6243:  URL_SCHEME_HTTP','LongInt'( 2);
6244:  URL_SCHEME_GOPHER','LongInt'( 3);
6245:  URL_SCHEME_MAILTO','LongInt'( 4);
6246:  URL_SCHEME_NEWS','LongInt'( 5);
6247:  URL_SCHEME_NNTP','LongInt'( 6);
```

```
6248:  URL_SCHEME_TELNET','LongInt'( 7);
6249:  URL_SCHEME_WAIS','LongInt'( 8);
6250:  URL_SCHEME_FILE','LongInt'( 9);
6251:  URL_SCHEME_MK','LongInt'( 10);
6252:  URL_SCHEME_HTTPS','LongInt'( 11);
6253:  URL_SCHEME_SHELL','LongInt'( 12);
6254:  URL_SCHEME_SNEWS','LongInt'( 13);
6255:  URL_SCHEME_LOCAL','LongInt'( 14);
6256:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6257:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6258:  URL_SCHEME_ABOUT','LongInt'( 17);
6259:  URL_SCHEME_RES','LongInt'( 18);
6260:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6261:  URL_SCHEME', 'Integer
6262:  URL_PART_NONE','LongInt'( 0);
6263:  URL_PART_SCHEME','LongInt'( 1);
6264:  URL_PART_HOSTNAME','LongInt'( 2);
6265:  URL_PART_USERNAME','LongInt'( 3);
6266:  URL_PART_PASSWORD','LongInt'( 4);
6267:  URL_PART_PORT','LongInt'( 5);
6268:  URL_PART_QUERY','LongInt'( 6);
6269:  URL_PART', 'DWORD
6270:  URLIS_URL','LongInt'( 0);
6271:  URLIS_OPAQUE','LongInt'( 1);
6272:  URLIS_NOHISTORY','LongInt'( 2);
6273:  URLIS_FILEURL','LongInt'( 3);
6274:  URLIS_APPLIABLE','LongInt'( 4);
6275:  URLIS_DIRECTORY','LongInt'( 5);
6276:  URLIS_HASQUERY','LongInt'( 6);
6277:  TUrlIs', 'DWORD
6278:  URL_UNESCAPE','LongWord( $10000000);
6279:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6280:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6281:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6282:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6283:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6284:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6285:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6286:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6287:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6288:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6289:  URL_INTERNAL_PATH','LongWord( $00800000);
6290:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6291:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6292:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6293:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6294:  URL_APPLY_DEFAULT','LongWord( $00000001);
6295:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6296:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6297:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6298:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6299:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6300:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6301:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6302:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6303:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6304:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6305:  Function UrlGetLocation( psz1 : PChar) : PChar
6306:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6307:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6308:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6309:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6310:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6311:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6312:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6313:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6314:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6315:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6316:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6317:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6318:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6319:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6320:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
       pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6321:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6322:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6323:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6324:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6325:  SHREGDEL_DEFAULT','LongWord( $00000000);
6326:  SHREGDEL_HKCU','LongWord( $00000001);
6327:  SHREGDEL_HKLM','LongWord( $00000010);
6328:  SHREGDEL_BOTH','LongWord( $00000011);
6329:  SHREGENUM_DEFAULT','LongWord( $00000000);
6330:  SHREGENUM_HKCU','LongWord( $00000001);
6331:  SHREGENUM_HKLM','LongWord( $00000010);
6332:  SHREGENUM_BOTH','LongWord( $00000011);
6333:  SHREGSET_HKCU','LongWord( $00000001);
6334:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6335:  SHREGSET_HKLM','LongWord( $00000004);
```

```
6336:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6337:  TSHRegDelFlags', 'DWORD
6338:  TSHRegEnumFlags', 'DWORD
6339:  HUSKEY', 'THandle
6340:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6341:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6342:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6343:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6344:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6345:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6346:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6347:  ASSOCF_VERIFY','LongWord( $00000040);
6348:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6349:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6350:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6351:  ASSOCF', 'DWORD
6352:  ASSOCSTR_COMMAND','LongInt'( 1);
6353:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6354:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6355:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6356:  ASSOCSTR_NOOPEN','LongInt'( 5);
6357:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6358:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6359:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6360:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6361:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6362:  ASSOCSTR_INFOTIP','LongInt'( 11);
6363:  ASSOCSTR_MAX','LongInt'( 12);
6364:  ASSOCSTR', 'DWORD
6365:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6366:  ASSOCKEY_APP','LongInt'( 2);
6367:  ASSOCKEY_CLASS','LongInt'( 3);
6368:  ASSOCKEY_BASECLASS','LongInt'( 4);
6369:  ASSOCKEY_MAX','LongInt'( 5);
6370:  ASSOCKEY', 'DWORD
6371:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6372:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6373:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6374:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6375:  ASSOCDATA_MAX','LongInt'( 5);
6376:  ASSOCDATA', 'DWORD
6377:  ASSOCENUM_NONE','LongInt'( 0);
6378:  ASSOCENUM', 'DWORD
6379:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6380:  SHACF_DEFAULT $00000000);
6381:  SHACF_FILESYSTEM','LongWord( $00000001);
6382:  SHACF_URLHISTORY','LongWord( $00000002);
6383:  SHACF_URLMRU','LongWord( $00000004);
6384:  SHACF_USETAB','LongWord( $00000008);
6385:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6386:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6387:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6388:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6389:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6390:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6391:  Procedure SHSetThreadRef( punk : IUnknown)
6392:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6393:  CTF_INSIST','LongWord( $00000001);
6394:  CTF_THREAD_REF','LongWord( $00000002);
6395:  CTF_PROCESS_REF','LongWord( $00000004);
6396:  CTF_COINIT','LongWord( $00000008);
6397:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6398:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6399:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6400:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6401:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6402:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6403:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6404:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6405:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6406:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6407:  Function SetRectEmpty( var lprc : TRect) : BOOL
6408:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6409:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6410:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6411:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6412:
6413:  Function InitializeFlatSB( hWnd : HWND) : Bool
6414:  Procedure UninitializeFlatSB( hWnd : HWND)
6415:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6416:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6417:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6418:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6419:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6420:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6421:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6422:
6423:
6424: //  ***************************************** 204 unit uPSI_ShellAPI;
```

```
6425: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6426: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6427: Procedure DragFinish( Drop : HDROP)
6428: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6429: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6430: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6431: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6432: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6433: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6434: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6435: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6436: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6437: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6438: Procedure SHFreeNameMappings( hNameMappings : THandle)
6439:
6440:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6441:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6442:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6443:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6444:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6445:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6446:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6447:  Function SimpleXMLEncode( const S : string) : string
6448:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6449:  Function XMLEncode( const S : string) : string
6450:  Function XMLDecode( const S : string) : string
6451:  Function EntityEncode( const S : string) : string
6452:  Function EntityDecode( const S : string) : string
6453:
6454: procedure RIRegister_CPort_Routines(S: TPSExec);
6455:  Procedure EnumComPorts( Ports : TStrings)
6456:  Procedure ListComPorts( Ports : TStrings)
6457:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6458:  Function GetComPorts: TStringlist;
6459:  Function StrToBaudRate( Str : string) : TBaudRate
6460:  Function StrToStopBits( Str : string) : TStopBits
6461:  Function StrToDataBits( Str : string) : TDataBits
6462:  Function StrToParity( Str : string) : TParityBits
6463:  Function StrToFlowControl( Str : string) : TFlowControl
6464:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6465:  Function StopBitsToStr( StopBits : TStopBits) : string
6466:  Function DataBitsToStr( DataBits : TDataBits) : string
6467:  Function ParityToStr( Parity : TParityBits) : string
6468:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6469:  Function ComErrorsToStr( Errors : TComErrors) : String
6470:
6471:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6472:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6473:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6474:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6475:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6476:  Function GetMessagePos : DWORD
6477:  Function GetMessageTime : Longint
6478:  Function GetMessageExtraInfo : Longint
6479:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6480:  Procedure JAddToRecentDocs( const Filename : string)
6481:  Procedure ClearRecentDocs
6482:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6483:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6484:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6485:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6486:  Function RecycleFile( FileToRecycle : string) : Boolean
6487:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6488:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6489:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6490:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6491:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6492:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6493:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6494:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6495:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6496:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6497:
6498: ****************************************** unit uPSI_JclPeImage;
6499:
6500:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6501: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6502:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6503:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
```

```
6504:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6505:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6506:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6507:  Function PeDoesExportFunction(const FileName:TFileName;const
       FuncName:string;Options:TJclSmartCompOptions):Bool;
6508:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
       ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6509:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
       FunctionName:string;Options:TJclSmartCompOptions):Bool
6510:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
       : string; Options : TJclSmartCompOptions) : Boolean
6511:  Function PeDoesImportLibrary(const FileName:TFileName;const
       LibraryName:string;Recursive:Boolean):Boolean);
6512:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
       Boolean; FullPathName : Boolean) : Boolean
6513:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
       LibraryName:string; IncludeLibNames : Boolean): Boolean
6514:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6515:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6516:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6517:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
       NamesList:TStrings):Bool
6518:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6519:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
       Descript:Bool):Bool;
6520:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6521:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6522:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6523:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6524:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6525:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6526:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
       PImageSectionHeader
6527:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6528:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6529:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
       ___Pointer;
6530:   SIRegister_TJclPeSectionStream(CL);
6531:   SIRegister_TJclPeMapImgHookItem(CL);
6532:   SIRegister_TJclPeMapImgHooks(CL);
6533:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
       NtHeaders:TImageNtHeaders):Boolean
6534:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6535:   Type TJclBorUmSymbolKind',(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6536:   TJclBorUmSymbolModifier', ( smQualified, smLinkProc )
6537:   TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6538:   TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6539:   TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6540:   TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6541:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
       TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6542:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6543:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6544:  Function PeBorUnmangleName3( const Name : string) : string;
6545:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6546:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6547:
6548:
6549: //****************** SysTools uPSI_StSystem; ************************************
6550:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6551:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6552:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6553:  //Procedure EnumerateDirectories(const
       StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6554:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
       IncludeItem:TIncludeItemFunc);
6555:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6556:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6557:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6558:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6559:  Function FlushOsBuffers( Handle : Integer) : Boolean
6560:  Function GetCurrentUser : AnsiString
6561:  Function GetDiskClass( Drive : Char) : DiskClass
6562:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
       SectorsPerCluster:Cardinal):Bool;
6563:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
       DiskSize:Double):Bool;
6564:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
       DiskSize:Comp):Boolean;
6565:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6566:  Function getDiskSpace2(const path: String; index: integer): int64;
6567:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6568:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6569:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6570:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6571:  Function GetLongPath( const APath : AnsiString) : AnsiString
6572:  Function GetMachineName : AnsiString
6573:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
```

```
6574:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6575:  Function GetShortPath( const APath : AnsiString) : AnsiString
6576:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6577:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6578:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6579:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6580:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6581:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6582:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6583:  Function IsDriveReady( Drive : Char) : Boolean
6584:  Function IsFile( const FileName : AnsiString) : Boolean
6585:  Function IsFileArchive( const S : AnsiString) : Integer
6586:  Function IsFileHidden( const S : AnsiString) : Integer
6587:  Function IsFileReadOnly( const S : AnsiString) : Integer
6588:  Function IsFileSystem( const S : AnsiString) : Integer
6589:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6590:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6591:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6592:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6593:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6594:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6595:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6596:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6597:  Function ValidDrive( Drive : Char) : Boolean
6598:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6599:
6600: //*****************************unit uPSI_JclLANMan;*****************************************
6601: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6602:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6603:  Function DeleteAccount( const Servername, Username : string) : Boolean
6604:  Function DeleteLocalAccount( Username : string) : Boolean
6605:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6606:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6607:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6608:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6609:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6610:  Function LocalGroupExists( const Group : string) : Boolean
6611:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6612:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6613:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6614:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6615:  Function IsLocalAccount( const AccountName : string) : Boolean
6616:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6617:  Function GetRandomString( NumChar : cardinal) : string
6618:
6619: //*****************************unit uPSI_cUtils;*****************************************
6620: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6621: Function cIsWinNT : boolean
6622:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6623:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6624:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6625:  Function cGetShortName( FileName : string) : string
6626:  Procedure cShowError( Msg : String)
6627:  Function cCommaStrToStr( s : string; formatstr : string) : string
6628:  Function cIncludeQuoteIfSpaces( s : string) : string
6629:  Function cIncludeQuoteIfNeeded( s : string) : string
6630:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6631:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6632:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6633:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6634:  Function cCodeInstoStr( s : string) : string
6635:  Function cStrtoCodeIns( s : string) : string
6636:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6637:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6638:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6639:  Function cPointtoStr( const pt : TPoint) : string
6640:  Function cListtoStr( const List : TStrings) : string
6641:  Function ListtoStr( const List : TStrings) : string
6642:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6643:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6644:  Function cGetFileTyp( const FileName : string) : TUnitType
6645:  Function cGetExTyp( const FileName : string) : TExUnitType
6646:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6647:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6648:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6649:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6650:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6651:  Function cGenMakePath( FileName : String) : String;
6652:  Function cGenMakePath2( FileName : String) : String
6653:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6654:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6655:  Function cCalcMod( Count : Integer) : Integer
6656:  Function cGetVersionString( FileName : string) : string
6657:  Function cCheckChangeDir( var Dir : string) : boolean
6658:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
```

```
6659:  Function cIsNumeric( s : string) : boolean
6660:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6661:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6662:  Function GetFileTyp( const FileName : string) : TUnitType
6663:  Function Atoi(const aStr: string): integer
6664:  Function Itoa(const aint: integer): string
6665:
6666:
6667: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6668: begin
6669:   FindClass('TOBJECT'),'EHTTP
6670:   FindClass('TOBJECT'),'EHTTPParser
6671:   //AnsiCharSet', 'set of AnsiChar
6672:   AnsiStringArray', 'array of AnsiString
6673:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6674:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6675:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6676:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6677:    +'CustomMinVersion : Integer; end
6678:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6679:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6680:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6681:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6682:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6683:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6684:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6685:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6686:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6687:    +'nection, hntOrigin, hntKeepAlive )
6688:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6689:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6690:    +' AnsiString; end
6691:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6692:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6693:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6694:   //PHTTPContentLength', '^THTTPContentLength // will not work
6695:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6696:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6697:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6698:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6699:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6700:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6701:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6702:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6703:    +' CustomStr : AnsiString; end
6704:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6705:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6706:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6707:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6708:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6709:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6710:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6711:    +'m; Custom : AnsiString; end
6712:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6713:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6714:    +' Custom : AnsiString; end
6715:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6716:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6717:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6718:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6719:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6720:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6721:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6722:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6723:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6724:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6725:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6726:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6727:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6728:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6729:    +'FieldEnum; List : array of THTTPContentEncoding; end
6730:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6731:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6732:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6733:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6734:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6735:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6736:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6737:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6738:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6739:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6740:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6741:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6742:    +'CustomFieldArray; Custom : AnsiString; end
6743:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6744:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6745:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6746:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6747:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
```

```
6748:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6749:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6750:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6751:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6752:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6753:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6754:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6755:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6756:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6757:   THTTPFixedHeaders','array[0..42] of AnsiString
6758:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6759:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6760:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6761:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6762:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6763:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6764:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6765:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6766:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6767:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6768:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6769:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6770:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6771:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6772:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6773:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6774:    +' THTTPDateField; Age : THTTPAgeField; end
6775:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6776:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6777:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6778:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6779:  Procedure InitHTTPRequest( var A : THTTPRequest)
6780:  Procedure InitHTTPResponse( var A : THTTPResponse)
6781:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6782:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6783:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6784:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6785:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6786:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6787:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6788:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6789:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6790:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6791:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6792:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6793:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6794:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6795:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6796:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6797:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6798:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6799:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6800:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6801:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6802:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6803:   THTTPStringOption', '( hsoNone )
6804:   THTTPStringOptions', 'set of THTTPStringOption
6805:   FindClass('TOBJECT'),'TAnsiStringBuilder
6806:
6807:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6808:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6809:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6810:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6811:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6812:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6813:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6814:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6815:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6816:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6817:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6818:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6819:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6820:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6821:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6822:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
```

```
6823:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6824:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6825:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions)
6826:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6827:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6828:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
       : THTTPStringOptions)
6829:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions);
6830:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
       THTTPStringOptions)
6831:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
       const P : THTTPStringOptions)
6832:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
       P:THTTPStringOptions);
6833:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
       THTTPStringOptions)
6834:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
       TAnsiStringBuilder; const P : THTTPStringOptions)
6835:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
       THTTPStrOptions);
6836:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
       P:THTTPStringOptions);
6837:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
       P:THTTPStringOptions);
6838:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6839:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6840:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6841:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6842:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6843:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6844:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
       Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6845:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6846:   SIRegister_THTTPParser(CL);
6847:   FindClass('TOBJECT'),'THTTPContentDecoder
6848:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6849:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6850:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6851:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6852:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6853:   SIRegister_THTTPContentDecoder(CL);
6854:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6855:   FindClass('TOBJECT'),'THTTPContentReader
6856:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6857:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
       LogLevel:Int;
6858:   SIRegister_THTTPContentReader(CL);
6859:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6860:   FindClass('TOBJECT'),'THTTPContentWriter
6861:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6862:   SIRegister_THTTPContentWriter(CL);
6863:  Procedure SelfTestcHTTPUtils
6864: end;
6865:
6866: (*----------------------------------------------------------------------------*)
6867: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6868: begin
6869:  'TLSLibraryVersion','String '1.00
6870:  'TLSError_None','LongInt'( 0);
6871:  'TLSError_InvalidBuffer','LongInt'( 1);
6872:  'TLSError_InvalidParameter','LongInt'( 2);
6873:  'TLSError_InvalidCertificate','LongInt'( 3);
6874:  'TLSError_InvalidState','LongInt'( 4);
6875:  'TLSError_DecodeError','LongInt'( 5);
6876:  'TLSError_BadProtocol','LongInt'( 6);
6877:  Function TLSErrorMessage( const TLSError : Integer) : String
6878:   SIRegister_ETLSError(CL);
6879:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6880:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6881:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6882:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6883:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6884:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6885:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6886:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6887:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6888:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6889:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6890:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6891:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6892:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6893:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6894:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
```

```
6895:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6896:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6897:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6898:   PTLSRandom', '^TTLSRandom // will not work
6899:  Procedure InitTLSRandom( var Random : TTLSRandom)
6900:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6901:  'TLSSessionIDMaxLen','LongInt'( 32);
6902:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6903:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6904:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6905:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6906:    +'; Signature : TTLSSignatureAlgorithm; end
6907:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6908:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6909:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6910:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6911:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6912:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6913:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6914:    +'nteger; Supported : Boolean; end
6915:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6916:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
6917:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6918:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6919:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6920:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6921:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6922:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6923:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6924:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6925:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
6926:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
6927:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6928:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6929:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
6930:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6931:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6932:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6933:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
6934:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6935:    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6936:    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6937:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
      IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6938:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
      ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6939:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
6940:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
6941:  Procedure SelfTestcTLSUtils
6942: end;
6943:
6944: (*----------------------------------------------------------------------------*)
6945: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6946: begin
6947:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
      integer; disks : integer; mx : integer; my : integer; end
6948:  // pBoard', '^tBoard // will not work
6949:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6950:  Function rCheckMove( color : byte; cx, cy : integer) : integer
6951:  //Function rDoStep( data : pBoard) : word
6952:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6953: end;
6954:
6955: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6956: begin
6957: Function InEditMode( ADataset : TDataset) : Boolean
6958: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6959: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6960: Function GetFieldText( AField : TField) : String
6961: end;
6962:
6963: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6964: begin
6965:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6966:   TMyPrintRange', '( prAll, prSelected )
6967:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6968:    +'ded, ssDateTime, ssTime, ssCustom )
6969:   TSortDirection', '( sdAscending, sdDescending )
6970:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6971:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
6972:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6973:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6974:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
```

```
6975:   SIRegister_TSortOptions(CL);
6976:   SIRegister_TPrintOptions(CL);
6977:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6978:   SIRegister_TSortedList(CL);
6979:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6980:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6981:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6982:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
6983:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
6984:   SIRegister_TFontSetting(CL);
6985:   SIRegister_TFontList(CL);
6986:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
6987:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
6988:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6989:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6990:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
6991:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
6992:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
6993:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
6994:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
6995:    +'r; var SortStyle : TSortStyle)
6996:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
6997:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
6998:   SIRegister_TSortGrid(CL);
6999:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
7000:  Function NormalCompare( const Str1, Str2 : String) : Integer
7001:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
7002:  Function NumericCompare( const Str1, Str2 : String) : Integer
7003:  Function TimeCompare( const Str1, Str2 : String) : Integer
7004:  //Function Compare( Item1, Item2 : Pointer) : Integer
7005: end;
7006:
7007: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
7008:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
7009:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7010:  Procedure IBDataBaseError
7011:  Function StatusVector : PISC_STATUS
7012:  Function StatusVectorArray : PStatusVector
7013:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7014:  Function StatusVectorAsText : string
7015:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7016:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7017:
7018:
7019: //****************************unit uPSI_BoldUtils;****************************************
7020:  Function CharCount( c : char; const s : string) : integer
7021:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7022:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7023:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7024:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7025:  Function BoldTrim( const S : string) : string
7026:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7027:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7028:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7029:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7030:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7031:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7032:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7033:  Function CapitalisedToSpaced( Capitalised : String) : String
7034:  Function SpacedToCapitalised( Spaced : String) : String
7035:  Function BooleanToString( BoolValue : Boolean) : String
7036:  Function StringToBoolean( StrValue : String) : Boolean
7037:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7038:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7039:  Function StringListToVarArray( List : TStringList) : variant
7040:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7041:  Function GetComputerNameStr : string
7042:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7043:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7044:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7045:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7046:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7047:  Procedure EnsureTrailing( var Str : String; ch : char)
7048:  Function BoldDirectoryExists( const Name : string) : Boolean
7049:  Function BoldForceDirectories( Dir : string) : Boolean
7050:  Function BoldRootRegistryKey : string
7051:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7052:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7053:  Function LogicalAnd( A, B : Integer) : Boolean
7054:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7055:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7056:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7057:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7058:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7059:  Function IsFirstInstance : Boolean
7060:  Procedure ActivateFirst( AString : PChar)
7061:  Procedure ActivateFirstCommandLine
7062:  function MakeAckPkt(const BlockNumber: Word): string;
```

```
7063:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7064:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7065:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7066:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7067:  function IdStrToWord(const Value: String): Word;
7068:  function IdWordToStr(const Value: Word): WordStr;
7069:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7070:  Function CPUFeatures : TCPUFeatures
7071:
7072: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7073: begin
7074:   AddTypeS('TXRTLBitIndex', 'Integer
7075:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7076:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7077:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7078:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7079:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7080:  Function XRTLSwapHiLo16( X : Word) : Word
7081:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7082:  Function XRTLSwapHiLo64( X : Int64) : Int64
7083:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7084:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7085:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7086:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7087:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7088:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7089:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7090:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7091:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7092:  Function XRTLPopulation( A : Cardinal) : Cardinal
7093: end;
7094:
7095: Function XRTLURLDecode( const ASrc : WideString) : WideString
7096: Function XRTLURLEncode( const ASrc : WideString) : WideString
7097: Function XRTLURINormalize( const AURI : WideString) : WideString
7098: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7099: Function XRTLExtractLongPathName(APath: string): string;
7100:
7101: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7102: begin
7103:   AddTypeS('Int8', 'ShortInt
7104:   AddTypeS('Int16', 'SmallInt
7105:   AddTypeS('Int32', 'LongInt
7106:   AddTypeS('UInt8', 'Byte
7107:   AddTypeS('UInt16', 'Word
7108:   AddTypeS('UInt32', 'LongWord
7109:   AddTypeS('UInt64', 'Int64
7110:   AddTypeS('Word8', 'UInt8
7111:   AddTypeS('Word16', 'UInt16
7112:   AddTypeS('Word32', 'UInt32
7113:   AddTypeS('Word64', 'UInt64
7114:   AddTypeS('LargeInt', 'Int64
7115:   AddTypeS('NativeInt', 'Integer
7116:   AddTypeS('NativeUInt', 'Cardinal
7117:   Const('BitsPerByte','LongInt'( 8);
7118:   Const('BitsPerWord','LongInt'( 16);
7119:   Const('BitsPerLongWord','LongInt'( 32);
7120:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7121:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7122:  Function MinI( const A, B : Integer) : Integer
7123:  Function MaxI( const A, B : Integer) : Integer
7124:  Function MinC( const A, B : Cardinal) : Cardinal
7125:  Function MaxC( const A, B : Cardinal) : Cardinal
7126:  Function SumClipI( const A, I : Integer) : Integer
7127:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7128:  Function InByteRange( const A : Int64) : Boolean
7129:  Function InWordRange( const A : Int64) : Boolean
7130:  Function InLongWordRange( const A : Int64) : Boolean
7131:  Function InShortIntRange( const A : Int64) : Boolean
7132:  Function InSmallIntRange( const A : Int64) : Boolean
7133:  Function InLongIntRange( const A : Int64) : Boolean
7134:   AddTypeS('Bool8', 'ByteBool
7135:   AddTypeS('Bool16', 'WordBool
7136:   AddTypeS('Bool32', 'LongBool
7137:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7138:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7139:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7140:  Const('MinSingle','Single').setExtended( 1.5E-45);
7141:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7142:  Const('MinDouble','Double').setExtended( 5.0E-324);
7143:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7144:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7145:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7146:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7147:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7148:  Function MinF( const A, B : Float) : Float
7149:  Function MaxF( const A, B : Float) : Float
7150:  Function ClipF( const Value : Float; const Low, High : Float) : Float
```

```
7151:  Function InSingleRange( const A : Float) : Boolean
7152:  Function InDoubleRange( const A : Float) : Boolean
7153:  Function InCurrencyRange( const A : Float) : Boolean;
7154:  Function InCurrencyRange1( const A : Int64) : Boolean;
7155:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7156:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7157:  Function FloatIsInfinity( const A : Extended) : Boolean
7158:  Function FloatIsNaN( const A : Extended) : Boolean
7159:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7160:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7161:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7162:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7163:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7164:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7165:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7166:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7167:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7168:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7169:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7170:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7171:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7172:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7173:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7174:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7175:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7176:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7177:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7178:  Function SetBitScanForward( const Value : LongWord) : Integer;
7179:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7180:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7181:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7182:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7183:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7184:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7185:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7186:  Function cReverseBits( const Value : LongWord) : LongWord;
7187:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7188:  Function cSwapEndian( const Value : LongWord) : LongWord
7189:  Function cTwosComplement( const Value : LongWord) : LongWord
7190:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7191:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7192:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7193:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7194:  Function cBitCount( const Value : LongWord) : LongWord
7195:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7196:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7197:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7198:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7199:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7200:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7201:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7202:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7203:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7204: //  AddTypeS('CharSet', 'set of AnsiChar
7205:   AddTypeS('CharSet', 'set of Char       //!!!
7206:   AddTypeS('AnsiCharSet', 'TCharSet
7207:   AddTypeS('ByteSet', 'set of Byte
7208:   AddTypeS('AnsiChar', 'Char
7209:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7210:  Function AsByteSet( const C : array of Byte) : ByteSet
7211:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7212:  Procedure ClearCharSet( var C : CharSet)
7213:  Procedure FillCharSet( var C : CharSet)
7214:  Procedure ComplementCharSet( var C : CharSet)
7215:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7216:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7217:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7218:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7219:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7220:  Function IsSubSet( const A, B : CharSet) : Boolean
7221:  Function IsEqual( const A, B : CharSet) : Boolean
7222:  Function IsEmpty( const C : CharSet) : Boolean
7223:  Function IsComplete( const C : CharSet) : Boolean
7224:  Function cCharCount( const C : CharSet) : Integer
7225:  Procedure ConvertCaseInsensitive( var C : CharSet)
7226:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7227:  Function IntRangeLength( const Low, High : Integer) : Int64
7228:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7229:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7230:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7231:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7232:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7233:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7234:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7235:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7236:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7237:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7238:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7239:   AddTypeS('UnicodeChar', 'WideChar
```

```
7240:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7241:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7242:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7243:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7244:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7245:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7246:  Function cSgn( const A : LongInt) : Integer;
7247:  Function cSgn1( const A : Int64) : Integer;
7248:  Function cSgn2( const A : Extended) : Integer;
7249:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7250:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7251:  Function WideCharToInt( const A : WideChar) : Integer
7252:  Function CharToInt( const A : Char) : Integer
7253:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7254:  Function IntToWideChar( const A : Integer) : WideChar
7255:  Function IntToChar( const A : Integer) : Char
7256:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7257:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7258:  Function IsHexChar( const Ch : Char) : Boolean
7259:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7260:  Function HexWideCharToInt( const A : WideChar) : Integer
7261:  Function HexCharToInt( const A : Char) : Integer
7262:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7263:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7264:  Function IntToUpperHexChar( const A : Integer) : Char
7265:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7266:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7267:  Function IntToLowerHexChar( const A : Integer) : Char
7268:  Function IntToStringA( const A : Int64) : AnsiString
7269:  Function IntToStringW( const A : Int64) : WideString
7270:  Function IntToString( const A : Int64) : String
7271:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7272:  Function UIntToStringW( const A : NativeUInt) : WideString
7273:  Function UIntToString( const A : NativeUInt) : String
7274:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7275:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7276:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7277:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7278:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7279:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7280:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7281:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7282:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7283:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7284:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7285:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7286:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7287:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7288:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7289:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7290:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7291:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7292:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7293:  Function StringToInt64A( const S : AnsiString) : Int64
7294:  Function StringToInt64W( const S : WideString) : Int64
7295:  Function StringToInt64( const S : String) : Int64
7296:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7297:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7298:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7299:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7300:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7301:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7302:  Function StringToIntA( const S : AnsiString) : Integer
7303:  Function StringToIntW( const S : WideString) : Integer
7304:  Function StringToInt( const S : String) : Integer
7305:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7306:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7307:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7308:  Function StringToLongWordA( const S : AnsiString) : LongWord
7309:  Function StringToLongWordW( const S : WideString) : LongWord
7310:  Function StringToLongWord( const S : String) : LongWord
7311:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7312:  Function HexToUIntW( const S : WideString) : NativeUInt
7313:  Function HexToUInt( const S : String) : NativeUInt
7314:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7315:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7316:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7317:  Function HexToLongWordA( const S : AnsiString) : LongWord
7318:  Function HexToLongWordW( const S : WideString) : LongWord
7319:  Function HexToLongWord( const S : String) : LongWord
7320:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7321:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7322:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7323:  Function OctToLongWordA( const S : AnsiString) : LongWord
7324:  Function OctToLongWordW( const S : WideString) : LongWord
7325:  Function OctToLongWord( const S : String) : LongWord
7326:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7327:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7328:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
```

```
7329:  Function BinToLongWordA( const S : AnsiString) : LongWord
7330:  Function BinToLongWordW( const S : WideString) : LongWord
7331:  Function BinToLongWord( const S : String) : LongWord
7332:  Function FloatToStringA( const A : Extended) : AnsiString
7333:  Function FloatToStringW( const A : Extended) : WideString
7334:  Function FloatToString( const A : Extended) : String
7335:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7336:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7337:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7338:  Function StringToFloatA( const A : AnsiString) : Extended
7339:  Function StringToFloatW( const A : WideString) : Extended
7340:  Function StringToFloat( const A : String) : Extended
7341:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7342:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7343:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7344:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
      PadChar: AnsiChar) : AnsiString
7345:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7346:  unit uPSI_cFundamentUtils;
7347:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7348:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7349:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7350:  Const('CCHARSET','Stringb64_XXEncode);
7351:  Const('CHEXSET','String'0123456789ABCDEF
7352:  Const('HEXDIGITS','String'0123456789ABCDEF
7353:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7354:  Const('DIGISET','String'0123456789
7355:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7356:  Const('DIGISET2','TCharset').SetSet('0123456789'
7357:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7358:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7359:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7360:  Const('NUMBERS','String'0123456789');
7361:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7362:  Function CharSetToStr( const C : CharSet) : AnsiString
7363:  Function StrToCharSet( const S : AnsiString) : CharSet
7364:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7365:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7366:  Function UUDecode( const S : AnsiString) : AnsiString
7367:  Function XXDecode( const S : AnsiString) : AnsiString
7368:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7369:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7370:  Function InterfaceToStrW( const I : IInterface) : WideString
7371:  Function InterfaceToStr( const I : IInterface) : String
7372:  Function ObjectClassName( const O : TObject) : String
7373:  Function ClassClassName( const C : TClass) : String
7374:  Function ObjectToStr( const O : TObject) : String
7375:  Function ObjectToString( const O : TObject) : String
7376:  Function CharSetToStr( const C : CharSet) : AnsiString
7377:  Function StrToCharSet( const S : AnsiString) : CharSet
7378:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7379:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7380:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7381:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7382:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7383:  Const('Bytes1KB','LongInt'( 1024);
7384:   SIRegister_IInterface(CL);
7385:  Procedure SelfTestCFundamentUtils
7386:
7387: Function CreateSchedule : IJclSchedule
7388: Function NullStamp : TTimeStamp
7389: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7390: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7391: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7392:
7393: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7394: begin
7395:  AddTypeS('TFunc', 'function(X : Float) : Float;
7396: Function InitGraphics( Width, Height : Integer) : Boolean
7397: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7398: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7399: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7400: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7401: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7402: Procedure SetGraphTitle( Title : String)
7403: Procedure SetOxTitle( Title : String)
7404: Procedure SetOyTitle( Title : String)
7405: Function GetGraphTitle : String
7406: Function GetOxTitle : String
7407: Function GetOyTitle : String
7408: Procedure PlotOxAxis( Canvas : TCanvas)
7409: Procedure PlotOyAxis( Canvas : TCanvas)
7410: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7411: Procedure WriteGraphTitle( Canvas : TCanvas)
7412: Function SetMaxCurv( NCurv : Byte) : Boolean
7413: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
```

```
7414: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7415: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7416: Procedure SetCurvStep( CurvIndex, Step : Integer)
7417: Function GetMaxCurv : Byte
7418: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7419: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7420: Function GetCurvLegend( CurvIndex : Integer) : String
7421: Function GetCurvStep( CurvIndex : Integer) : Integer
7422: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7423: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7424: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7425: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7426: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7427: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7428: Function Xpixel( X : Float) : Integer
7429: Function Ypixel( Y : Float) : Integer
7430: Function Xuser( X : Integer) : Float
7431: Function Yuser( Y : Integer) : Float
7432: end;
7433:
7434: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7435: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7436: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7437: Procedure FFT_Integer_Cleanup
7438: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7439: //unit uPSI_JclStreams;
7440: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7441: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7442: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7443: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7444:
7445: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7446: begin
7447:  FindClass('TOBJECT'),'EInvalidDest
7448:  FindClass('TOBJECT'),'EFCantMove
7449:  Procedure fmxCopyFile( const FileName, DestName : string)
7450:  Procedure fmxMoveFile( const FileName, DestName : string)
7451:  Function fmxGetFileSize( const FileName : string) : LongInt
7452:  Function fmxFileDateTime( const FileName : string) : TDateTime
7453:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7454:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7455: end;
7456:
7457: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7458: begin
7459:  SIRegister_IFindFileIterator(CL);
7460:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7461: end;
7462:
7463: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7464: begin
7465:  Function SkipWhite( cp : PChar) : PChar
7466:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7467:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7468:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7469: end;
7470:
7471: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7472: begin
7473:   SIRegister_TStringHashMapTraits(CL);
7474:  Function CaseSensitiveTraits : TStringHashMapTraits
7475:  Function CaseInsensitiveTraits : TStringHashMapTraits
7476:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7477:    +'e; Right : PHashNode; end
7478:   //PHashArray', '^THashArray // will not work
7479:   SIRegister_TStringHashMap(CL);
7480:  THashValue', 'Cardinal
7481:  Function StrHash( const s : string) : THashValue
7482:  Function TextHash( const s : string) : THashValue
7483:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7484:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7485:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7486:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7487:   SIRegister_TCaseSensitiveTraits(CL);
7488:   SIRegister_TCaseInsensitiveTraits(CL);
7489:
7490:
7491: //************************************************************************unit uPSI_umath;
7492:  Function uExpo( X : Float) : Float
7493:  Function uExp2( X : Float) : Float
7494:  Function uExp10( X : Float) : Float
7495:  Function uLog( X : Float) : Float
7496:  Function uLog2( X : Float) : Float
7497:  Function uLog10( X : Float) : Float
7498:  Function uLogA( X, A : Float) : Float
7499:  Function uIntPower( X : Float; N : Integer): Float
7500:  Function uPower( X, Y : Float) : Float
7501:  Function SgnGamma( X : Float) : Integer
7502:  Function Stirling( X : Float) : Float
```

```
7503:  Function StirLog( X : Float) : Float
7504:  Function Gamma( X : Float) : Float
7505:  Function LnGamma( X : Float) : Float
7506:  Function DiGamma( X : Float) : Float
7507:  Function TriGamma( X : Float) : Float
7508:  Function IGamma( X : Float) : Float
7509:  Function JGamma( X : Float) : Float
7510:  Function InvGamma( X : Float) : Float
7511:  Function Erf( X : Float) : Float
7512:  Function Erfc( X : Float) : Float
7513: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7514: { Correlation coefficient between samples X and Y }
7515: function DBeta(A, B, X : Float) : Float;
7516: { Density of Beta distribution with parameters A and B }
7517: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7518: Function Beta(X, Y : Float) : Float
7519: Function Binomial( N, K : Integer) : Float
7520: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7521: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7522: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7523: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7524: Function DNorm( X : Float) : Float
7525:
7526: function DGamma(A, B, X : Float) : Float;
7527: { Density of Gamma distribution with parameters A and B }
7528: function DKhi2(Nu : Integer; X : Float) : Float;
7529: { Density of Khi-2 distribution with Nu d.o.f. }
7530: function DStudent(Nu : Integer; X : Float) : Float;
7531: { Density of Student distribution with Nu d.o.f. }
7532: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7533: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7534: function IBeta(A, B, X : Float) : Float;
7535: { Incomplete Beta function}
7536: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7537:
7538: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7539: begin
7540:  Procedure SetOptAlgo( Algo : TOptAlgo)
7541: procedure SetOptAlgo(Algo : TOptAlgo);
7542: { ----------------------------------------------------------------
7543:   Sets the optimization algorithm according to Algo, which must be
7544:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7545:
7546:  Function GetOptAlgo : TOptAlgo
7547:  Procedure SetMaxParam( N : Byte)
7548:  Function GetMaxParam : Byte
7549:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7550:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7551:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7552:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
     Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7553:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
     MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7554:  Procedure SetMCFile( FileName : String)
7555:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7556:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
     LastPar:Integer;V:TMatrix);
7557: end;
7558:
7559: (*----------------------------------------------------------------------------*)
7560: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7561: begin
7562:  Procedure SaveSimplex( FileName : string)
7563:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7564: end;
7565: (*----------------------------------------------------------------------------*)
7566: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7567: begin
7568:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7569:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7570: end;
7571:
7572: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7573: begin
7574:  Function LTrim( S : String) : String
7575:  Function RTrim( S : String) : String
7576:  Function uTrim( S : String) : String
7577:  Function StrChar( N : Byte; C : Char) : String
7578:  Function RFill( S : String; L : Byte) : String
7579:  Function LFill( S : String; L : Byte) : String
7580:  Function CFill( S : String; L : Byte) : String
7581:  Function Replace( S : String; C1, C2 : Char) : String
7582:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7583:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7584:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7585:  Function FloatStr( X : Float) : String
7586:  Function IntStr( N : LongInt) : String
7587:  Function uCompStr( Z : Complex) : String
7588: end;
```

```
7589:
7590: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7591: begin
7592:  Function uSinh( X : Float) : Float
7593:  Function uCosh( X : Float) : Float
7594:  Function uTanh( X : Float) : Float
7595:  Function uArcSinh( X : Float) : Float
7596:  Function uArcCosh( X : Float) : Float
7597:  Function ArcTanh( X : Float) : Float
7598:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7599: end;
7600:
7601: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7602: begin
7603: type RNG_Type =
7604:   (RNG_MWC,      { Multiply-With-Carry }
7605:    RNG_MT,       { Mersenne Twister }
7606:    RNG_UVAG);    { Universal Virtual Array Generator }
7607:  Procedure SetRNG( RNG : RNG_Type)
7608:  Procedure InitGen( Seed : RNG_IntType)
7609:  Procedure SRand( Seed : RNG_IntType)
7610:  Function IRanGen : RNG_IntType
7611:  Function IRanGen31 : RNG_IntType
7612:  Function RanGen1 : Float
7613:  Function RanGen2 : Float
7614:  Function RanGen3 : Float
7615:  Function RanGen53 : Float
7616: end;
7617:
7618: //  Optimization by Simulated Annealing
7619: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7620: begin
7621:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7622:  Procedure SA_CreateLogFile( FileName : String)
7623:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7624: end;
7625:
7626: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7627: begin
7628:  Procedure InitUVAGbyString( KeyPhrase : string)
7629:  Procedure InitUVAG( Seed : RNG_IntType)
7630:  Function IRanUVAG : RNG_IntType
7631: end;
7632:
7633: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7634: begin
7635:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7636:  Procedure GA_CreateLogFile( LogFileName : String)
7637:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7638: end;
7639:
7640:  TVector', 'array of Float
7641: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7642: begin
7643:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7644:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7645: end;
7646:
7647: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7648: begin
7649:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7650:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7651: end;
7652:
7653: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7654: begin
7655:   FT_Result', 'Integer
7656:   //TDWordptr', '^DWord // will not work
7657:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7658:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7659:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7660:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7661:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7662:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7663:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7664:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7665:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7666:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7667:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7668:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7669:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7670:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7671:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7672:    yte; end
7673: end;
7674:
7675:
7676: //*********************************** PaintFX*************************
7677: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
```

```
7678: begin
7679:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7680:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7681:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7682:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7683:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7684:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7685:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7686:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7687:     Procedure Turn( Src, Dst : TBitmap)
7688:     Procedure TurnRight( Src, Dst : TBitmap)
7689:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7690:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7691:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7692:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7693:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7694:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7695:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7696:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7697:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7698:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7699:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7700:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7701:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7702:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7703:     Procedure Emboss( var Bmp : TBitmap)
7704:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7705:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7706:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7707:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7708:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7709:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7710:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7711:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7712:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7713:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7714:     Procedure SemiOpaque( Src, Dst : TBitmap)
7715:     Procedure ShadowDownLeft( const Dst : TBitmap)
7716:     Procedure ShadowDownRight( const Dst : TBitmap)
7717:     Procedure ShadowUpLeft( const Dst : TBitmap)
7718:     Procedure ShadowUpRight( const Dst : TBitmap)
7719:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7720:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7721:     Procedure FlipRight( const Dst : TBitmap)
7722:     Procedure FlipDown( const Dst : TBitmap)
7723:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7724:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7725:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7726:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7727:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7728:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7729:     Procedure SmoothResize( var Src, Dst : TBitmap)
7730:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7731:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7732:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7733:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7734:     Procedure GrayScale( const Dst : TBitmap)
7735:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7736:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7737:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7738:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7739:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7740:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7741:     Procedure AntiAlias( const Dst : TBitmap)
7742:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7743:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7744:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7745:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7746:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7747:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7748:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7749:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7750:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7751:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7752:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7753:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7754:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7755:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7756:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7757:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7758:     Procedure Invert( Src : TBitmap)
7759:     Procedure MirrorRight( Src : TBitmap)
7760:     Procedure MirrorDown( Src : TBitmap)
7761:   end;
7762: end;
7763:
7764: (*----------------------------------------------------------------------------*)
7765: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7766: begin
```

```
7767:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7768:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7769:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7770:   SIRegister_TJvPaintFX(CL);
7771:  Function SplineFilter( Value : Single) : Single
7772:  Function BellFilter( Value : Single) : Single
7773:  Function TriangleFilter( Value : Single) : Single
7774:  Function BoxFilter( Value : Single) : Single
7775:  Function HermiteFilter( Value : Single) : Single
7776:  Function Lanczos3Filter( Value : Single) : Single
7777:  Function MitchellFilter( Value : Single) : Single
7778: end;
7779:
7780:
7781: (*----------------------------------------------------------------------------*)
7782: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7783: begin
7784:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7785:  TeeMsg_DefaultSeriesName','String 'Series
7786:  TeeMsg_DefaultToolName','String 'ChartTool
7787:  ChartComponentPalette','String 'TeeChart
7788:  TeeMaxLegendColumns',LongInt'( 2);
7789:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7790:  TeeTitleFootDistance,LongInt( 5);
7791:   SIRegister_TCustomChartWall(CL);
7792:   SIRegister_TChartWall(CL);
7793:   SIRegister_TChartLegendGradient(CL);
7794:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7795:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7796:   FindClass('TOBJECT'),'LegendException
7797:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7798:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7799:   FindClass('TOBJECT'),'TCustomChartLegend
7800:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7801:   TLegendSymbolPosition', '( spLeft, spRight )
7802:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7803:   TSymbolCalcHeight', 'Function  : Integer
7804:   SIRegister_TLegendSymbol(CL);
7805:   SIRegister_TTeeCustomShapePosition(CL);
7806:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7807:   SIRegister_TLegendTitle(CL);
7808:   SIRegister_TLegendItem(CL);
7809:   SIRegister_TLegendItems(CL);
7810:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7811:   FindClass('TOBJECT'),'TCustomChart
7812:   SIRegister_TCustomChartLegend(CL);
7813:   SIRegister_TChartLegend(CL);
7814:   SIRegister_TChartTitle(CL);
7815:   SIRegister_TChartFootTitle(CL);
7816:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7817:    +'eButton; Shift : TShiftState; X, Y : Integer)
7818:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7819:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7820:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7821:    +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7822:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7823:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7824:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7825:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7826:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7827:    +'toMax : Boolean; Min : Double; Max : Double; end
7828:   TAllAxisSavedScales', 'array of TAxisSavedScales
7829:   SIRegister_TChartBackWall(CL);
7830:   SIRegister_TChartRightWall(CL);
7831:   SIRegister_TChartBottomWall(CL);
7832:   SIRegister_TChartLeftWall(CL);
7833:   SIRegister_TChartWalls(CL);
7834:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7835:   SIRegister_TCustomChart(CL);
7836:   SIRegister_TChart(CL);
7837:   SIRegister_TTeeSeriesTypes(CL);
7838:   SIRegister_TTeeToolTypes(CL);
7839:   SIRegister_TTeeDragObject(CL);
7840:   SIRegister_TColorPalettes(CL);
7841:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7842:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7843:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7844:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7845:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7846:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7847:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7848:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7849:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7850:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7851:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
```

```
7852:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7853:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7854:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7855:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7856:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7857:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7858:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7859:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7860:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7861:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
      R:TRect;ReferenceChart:TCustomChart);
7862:   SIRegister_TChartTheme(CL);
7863:   //TChartThemeClass', 'class of TChartTheme
7864:   //TCanvasClass', 'class of TCanvas3D
7865:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7866:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7867:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7868:  Procedure ShowMessageUser( const S : String)
7869:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7870:  Function HasLabels( ASeries : TChartSeries) : Boolean
7871:  Function HasColors( ASeries : TChartSeries) : Boolean
7872:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7873:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7874: end;
7875:
7876:
7877: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7878: begin
7879:  //'TeeFormBorderStyle',' bsNone);
7880:   SIRegister_TMetafile(CL);
7881:  'TeeDefVerticalMargin','LongInt'( 4);
7882:  'TeeDefHorizMargin','LongInt'( 3);
7883:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7884:  'TeeMsg_TeeHand','String 'crTeeHand
7885:  'TeeNormalPrintDetail','LongInt'( 0);
7886:  'TeeHighPrintDetail','LongInt'( - 100);
7887:  'TeeDefault_PrintMargin','LongInt'( 15);
7888:  'MaxDefaultColors','LongInt'( 19);
7889:  'TeeTabDelimiter','Char #9);
7890:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7891:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7892:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7893:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7894:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7895:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7896:   SIRegister_TCustomPanelNoCaption(CL);
7897:   FindClass('TOBJECT'),'TCustomTeePanel
7898:   SIRegister_TZoomPanning(CL);
7899:   SIRegister_TTeeEvent(CL);
7900:   //SIRegister_TTeeEventListeners(CL);
7901:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7902:   SIRegister_TTeeMouseEvent(CL);
7903:   SIRegister_TCustomTeePanel(CL);
7904:   //TChartGradient', 'TTeeGradient
7905:   //TChartGradientClass', 'class of TChartGradient
7906:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7907:   SIRegister_TTeeZoomPen(CL);
7908:   SIRegister_TTeeZoomBrush(CL);
7909:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7910:   SIRegister_TTeeZoom(CL);
7911:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7912:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7913:   SIRegister_TBackImage(CL);
7914:   SIRegister_TCustomTeePanelExtended(CL);
7915:   //TChartBrushClass', 'class of TChartBrush
7916:   SIRegister_TTeeCustomShapeBrushPen(CL);
7917:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7918:   TTextFormat', '( ttfNormal, ttfHtml )
7919:   SIRegister_TTeeCustomShape(CL);
7920:   SIRegister_TTeeShape(CL);
7921:   SIRegister_TTeeExportData(CL);
7922:  Function TeeStr( const Num : Integer) : String
7923:  Function DateTimeDefaultFormat( const AStep : Double) : String
7924:  Function TEEDaysInMonth( Year, Month : Word) : Word
7925:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7926:  Function NextDateTimeStep( const AStep : Double) : Double
7927:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7928:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7929:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7930:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7931:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7932:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7933:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7934:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7935:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7936:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7937:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7938:  Function DelphiToLocalFormat( const Format : String) : String
7939:  Function LocalToDelphiFormat( const Format : String) : String
```

```
7940:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
7941:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7942:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
       AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7943:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
7944:   TTeeSortSwap', 'Procedure ( a, b : Integer)
7945:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
7946:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7947:  Function TeeExtractField( St : String; Index : Integer) : String;
7948:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7949:  Function TeeNumFields( St : String) : Integer;
7950:  Function TeeNumFields1( const St, Separator : String) : Integer;
7951:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7952:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7953:  // TColorArray', 'array of TColor
7954:  Function GetDefaultColor( const Index : Integer) : TColor
7955:  Procedure SetDefaultColorPalette;
7956:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
7957:  'TeeCheckBoxSize','LongInt'( 11);
7958:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7959:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
7960:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
7961:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
7962:  Procedure TeeTranslateControl( AControl : TControl);
7963:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7964:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
7965:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
7966:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
7967:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
7968:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
7969:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
7970:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
7971:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
7972:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
7973:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
7974:  Procedure TeeSaveStringOption( const AKey, Value : String)
7975:  Function TeeDefaultXMLEncoding : String
7976:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
7977:   TeeWindowHandle', 'Integer
7978:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
7979:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
7980:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
7981: end;
7982:
7983:
7984: using mXBDEUtils
7985: ********************************************************************************
7986:  Procedure SetAlias( aAlias, aDirectory : String)
7987:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
7988:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
7989:  Procedure SetBDE( aPath, aNode, aValue : String)
7990:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
7991:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
7992:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
7993:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
7994:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
7995:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
7996:
7997:
7998: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
7999: begin
8000: AddClassN(FindClass('TOBJECT'),'EDateTime
8001: Function DatePart( const D : TDateTime) : Integer
8002: Function TimePart( const D : TDateTime) : Double
8003: Function Century( const D : TDateTime) : Word
8004: Function Year( const D : TDateTime) : Word
8005: Function Month( const D : TDateTime) : Word
8006: Function Day( const D : TDateTime) : Word
8007: Function Hour( const D : TDateTime) : Word
8008: Function Minute( const D : TDateTime) : Word
8009: Function Second( const D : TDateTime) : Word
8010: Function Millisecond( const D : TDateTime) : Word
8011: ('OneDay','Extended').setExtended( 1.0);
8012: ('OneHour','Extended').SetExtended( OneDay / 24);
8013: ('OneMinute','Extended').SetExtended( OneHour / 60);
8014: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8015: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8016: ('OneWeek','Extended').SetExtended( OneDay * 7);
8017: ('HoursPerDay','Extended').SetExtended( 24);
8018: ('MinutesPerHour','Extended').SetExtended( 60);
8019: ('SecondsPerMinute','Extended').SetExtended( 60);
8020: Procedure SetYear( var D : TDateTime; const Year : Word)
8021: Procedure SetMonth( var D : TDateTime; const Month : Word)
8022: Procedure SetDay( var D : TDateTime; const Day : Word)
8023: Procedure SetHour( var D : TDateTime; const Hour : Word)
8024: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8025: Procedure SetSecond( var D : TDateTime; const Second : Word)
8026: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
```

```
8027: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8028: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8029: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8030: Function IsAM( const D : TDateTime) : Boolean
8031: Function IsPM( const D : TDateTime) : Boolean
8032: Function IsMidnight( const D : TDateTime) : Boolean
8033: Function IsNoon( const D : TDateTime) : Boolean
8034: Function IsSunday( const D : TDateTime) : Boolean
8035: Function IsMonday( const D : TDateTime) : Boolean
8036: Function IsTuesday( const D : TDateTime) : Boolean
8037: Function IsWedneday( const D : TDateTime) : Boolean
8038: Function IsThursday( const D : TDateTime) : Boolean
8039: Function IsFriday( const D : TDateTime) : Boolean
8040: Function IsSaturday( const D : TDateTime) : Boolean
8041: Function IsWeekend( const D : TDateTime) : Boolean
8042: Function Noon( const D : TDateTime) : TDateTime
8043: Function Midnight( const D : TDateTime) : TDateTime
8044: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8045: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8046: Function NextWorkday( const D : TDateTime) : TDateTime
8047: Function PreviousWorkday( const D : TDateTime) : TDateTime
8048: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8049: Function LastDayOfYear( const D : TDateTime) : TDateTime
8050: Function EasterSunday( const Year : Word) : TDateTime
8051: Function GoodFriday( const Year : Word) : TDateTime
8052: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8053: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8054: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8055: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8056: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8057: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8058: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8059: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8060: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8061: Function DayOfYear( const D : TDateTime) : Integer
8062: Function DaysInMonth( const Ye, Mo : Word) : Integer
8063: Function DaysInMonth( const D : TDateTime) : Integer
8064: Function DaysInYear( const Ye : Word) : Integer
8065: Function DaysInYearDate( const D : TDateTime) : Integer
8066: Function WeekNumber( const D : TDateTime) : Integer
8067: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8068: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8069: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8070: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8071: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8072: Function DiffHours( const D1, D2 : TDateTime) : Integer
8073: Function DiffDays( const D1, D2 : TDateTime) : Integer
8074: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8075: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8076: Function DiffYears( const D1, D2 : TDateTime) : Integer
8077: Function GMTBias : Integer
8078: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8079: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8080: Function NowAsGMTTime : TDateTime
8081: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8082: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8083: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8084: Function DateTimeToANSI( const D : TDateTime) : Integer
8085: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8086: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8087: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8088: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8089: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8090: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8091: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8092: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8093: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8094: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8095: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8096: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8097: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8098: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8099: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8100: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8101: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8102: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8103: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8104: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8105: Function EnglishShortMonthA( const S : AnsiString) : Integer
8106: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8107: Function EnglishLongMonthA( const S : AnsiString) : Integer
8108: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8109: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8110: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8111: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8112: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8113: Function RFCMonthA( const S : AnsiString) : Word
8114: Function RFCMonthU( const S : UnicodeString) : Word
8115: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
```

```
8116: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8117: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8118: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8119: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8120: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8121: Function NowAsRFCDateTimeA : AnsiString
8122: Function NowAsRFCDateTimeU : UnicodeString
8123: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8124: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8125: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8126: Function TimePeriodStr( const D : TDateTime) : AnsiString
8127: Procedure SelfTest
8128: end;
8129: //****************************************CFileUtils
8130: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8131: Function PathHasDriveLetter( const Path : String) : Boolean
8132: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8133: Function PathIsDriveLetter( const Path : String) : Boolean
8134: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8135: Function PathIsDriveRoot( const Path : String) : Boolean
8136: Function PathIsRootA( const Path : AnsiString) : Boolean
8137: Function PathIsRoot( const Path : String) : Boolean
8138: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8139: Function PathIsUNCPath( const Path : String) : Boolean
8140: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8141: Function PathIsAbsolute( const Path : String) : Boolean
8142: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8143: Function PathIsDirectory( const Path : String) : Boolean
8144: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8145: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8146: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8147: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8148: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8149: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8150: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8151: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8152: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8153: Function PathCanonical( const Path : String; const PathSep : Char) : String
8154: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8155: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8156: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8157: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8158: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8159: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8160: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8161: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8162: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8163: Function FileNameValid( const FileName : String) : String
8164: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8165: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8166: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8167: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8168: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8169: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8170: Procedure CCopyFile( const FileName, DestName : String)
8171: Procedure CMoveFile( const FileName, DestName : String)
8172: Function CDeleteFiles( const FileMask : String) : Boolean
8173: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8174: Procedure FileCloseEx( const FileHandle : TFileHandle)
8175: Function FileExistsA( const FileName : AnsiString) : Boolean
8176:  Function CFileExists( const FileName : String) : Boolean
8177:  Function CFileGetSize( const FileName : String) : Int64
8178: Function FileGetDateTime( const FileName : String) : TDateTime
8179: Function FileGetDateTime2( const FileName : String) : TDateTime
8180: Function FileIsReadOnly( const FileName : String) : Boolean
8181: Procedure FileDeleteEx( const FileName : String)
8182: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8183: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
      : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8184: Function DirectoryEntryExists( const Name : String) : Boolean
8185: Function DirectoryEntrySize( const Name : String) : Int64
8186:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8187: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8188: Procedure CDirectoryCreate( const DirectoryName : String)
8189: Function GetFirstFileNameMatching( const FileMask : String) : String
8190: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8191: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8192: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8193:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8194:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8195: Function DriveIsValid( const Drive : Char) : Boolean
8196: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8197: Function DriveFreeSpace( const Path : AnsiString) : Int64
8198:
8199: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8200: begin
8201:  AddClassN(FindClass('TOBJECT'),'ETimers
8202:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8203: Function TickDelta( const D1, D2 : LongWord) : Integer
```

```
8204: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8205:   AddTypeS('THPTimer', 'Int64
8206: Procedure StartTimer( var Timer : THPTimer)
8207: Procedure StopTimer( var Timer : THPTimer)
8208: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8209: Procedure InitStoppedTimer( var Timer : THPTimer)
8210: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8211: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8212: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8213: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8214: Function GetHighPrecisionFrequency : Int64
8215: Function GetHighPrecisionTimerOverhead : Int64
8216: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8217: Procedure SelfTestCTimer
8218: end;
8219:
8220: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8221: begin
8222:  Function RandomSeed : LongWord
8223:  Procedure AddEntropy( const Value : LongWord)
8224:  Function RandomUniform : LongWord;
8225:  Function RandomUniform1( const N : Integer) : Integer;
8226:  Function RandomBoolean : Boolean
8227:  Function RandomByte : Byte
8228:  Function RandomByteNonZero : Byte
8229:  Function RandomWord : Word
8230:  Function RandomInt64 : Int64;
8231:  Function RandomInt641( const N : Int64) : Int64;
8232:  Function RandomHex( const Digits : Integer) : String
8233:  Function RandomFloat : Extended
8234:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8235:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8236:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8237:  Function mwcRandomLongWord : LongWord
8238:  Function urnRandomLongWord : LongWord
8239:  Function moaRandomFloat : Extended
8240:  Function mwcRandomFloat : Extended
8241:  Function RandomNormalF : Extended
8242:  Procedure SelfTestCRandom
8243: end;
8244:
8245: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8246: begin
8247:  // PIntArray', '^TIntArray // will not work
8248:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8249:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8250:  Function synMax( x, y : integer) : integer
8251:  Function synMin( x, y : integer) : integer
8252:  Function synMinMax( x, mi, ma : integer) : integer
8253:  Procedure synSwapInt( var l, r : integer)
8254:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8255:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8256:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8257:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8258:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8259:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8260:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8261:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8262:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8263:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8264:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8265:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8266:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8267:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8268:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8269:  ('C3_NONSPACING','LongInt'( 1);
8270:  'C3_DIACRITIC','LongInt'( 2);
8271:  'C3_VOWELMARK','LongInt'( 4);
8272:  ('C3_SYMBOL','LongInt'( 8);
8273:  ('C3_KATAKANA','LongWord( $0010);
8274:  ('C3_HIRAGANA','LongWord( $0020);
8275:  ('C3_HALFWIDTH','LongWord( $0040);
8276:  ('C3_FULLWIDTH','LongWord( $0080);
8277:  ('C3_IDEOGRAPH','LongWord( $0100);
8278:  ('C3_KASHIDA','LongWord( $0200);
8279:  ('C3_LEXICAL','LongWord( $0400);
8280:  ('C3_ALPHA','LongWord( $8000);
8281:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8282:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8283:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8284:  Function synIsStringType( Value : Word) : TStringType
8285:  Function synGetEOL( Line : PChar) : PChar
8286:  Function synEncodeString( s : string) : string
8287:  Function synDecodeString( s : string) : string
8288:  Procedure synFreeAndNil( var Obj: TObject)
8289:  Procedure synAssert( Expr : Boolean)
8290:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8291:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8292:   TReplaceFlags', 'set of TReplaceFlag )
```

```
8293:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8294:  Function synGetRValue( RGBValue : TColor) : byte
8295:  Function synGetGValue( RGBValue : TColor) : byte
8296:  Function synGetBValue( RGBValue : TColor) : byte
8297:  Function synRGB( r, g, b : Byte) : Cardinal
8298: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8299:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8300:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8301:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8302:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8303: end;
8304:
8305:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8306:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8307:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8308:
8309: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8310: begin
8311:  Function STimeZoneBias : integer
8312:  Function TimeZone : string
8313:  Function Rfc822DateTime( t : TDateTime) : string
8314:  Function CDateTime( t : TDateTime) : string
8315:  Function SimpleDateTime( t : TDateTime) : string
8316:  Function AnsiCDateTime( t : TDateTime) : string
8317:  Function GetMonthNumber( Value : String) : integer
8318:  Function GetTimeFromStr( Value : string) : TDateTime
8319:  Function GetDateMDYFromStr( Value : string) : TDateTime
8320:  Function DecodeRfcDateTime( Value : string) : TDateTime
8321:  Function GetUTTime : TDateTime
8322:  Function SetUTTime( Newdt : TDateTime) : Boolean
8323:  Function SGetTick : LongWord
8324:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8325:  Function CodeInt( Value : Word) : Ansistring
8326:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8327:  Function CodeLongInt( Value : LongInt) : Ansistring
8328:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8329:  Function DumpStr( const Buffer : Ansistring) : string
8330:  Function DumpExStr( const Buffer : Ansistring) : string
8331:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8332:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8333:  Function TrimSPLeft( const S : string) : string
8334:  Function TrimSPRight( const S : string) : string
8335:  Function TrimSP( const S : string) : string
8336:  Function SeparateLeft( const Value, Delimiter : string) : string
8337:  Function SeparateRight( const Value, Delimiter : string) : string
8338:  Function SGetParameter( const Value, Parameter : string) : string
8339:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8340:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8341:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8342:  Function GetEmailAddr( const Value : string) : string
8343:  Function GetEmailDesc( Value : string) : string
8344:  Function CStrToHex( const Value : Ansistring) : string
8345:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8346:  Function CBinToInt( const Value : string) : Integer
8347:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8348:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8349:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8350:  Function CRPos( const Sub, Value : String) : Integer
8351:  Function FetchBin( var Value : string; const Delimiter : string) : string
8352:  Function CFetch( var Value : string; const Delimiter : string) : string
8353:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8354:  Function IsBinaryString( const Value : AnsiString) : Boolean
8355:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8356:  Procedure StringsTrim( const value : TStrings)
8357:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8358:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8359:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8360:  Function CCountOfChar( const Value : string; aChr : char) : integer
8361:  Function UnquoteStr( const Value : string; Quote : Char) : string
8362:  Function QuoteStr( const Value : string; Quote : Char) : string
8363:  Procedure HeadersToList( const Value : TStrings)
8364:  Procedure ListToHeaders( const Value : TStrings)
8365:  Function SwapBytes( Value : integer) : integer
8366:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8367:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8368:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8369:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8370:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8371:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8372: end;
8373:
8374: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8375: begin
8376:  ('CrcBufSize','LongInt'( 2048);
8377:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8378:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8379:  Function Adler32OfFile( FileName : AnsiString) : LongInt
```

```
8380:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8381:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8382:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8383:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8384:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8385:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8386:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8387:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8388:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8389:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8390:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8391:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8392: end;
8393:
8394: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8395: begin
8396:  function CreateOleObject(const ClassName: String): IDispatch;
8397:  function GetActiveOleObject(const ClassName: String): IDispatch;
8398:  function ProgIDToClassID(const ProgID: string): TGUID;
8399:  function ClassIDToProgID(const ClassID: TGUID): string;
8400:  function CreateClassID: string;
8401:  function CreateGUIDString: string;
8402:  function CreateGUIDID: string;
8403:  procedure OleError(ErrorCode: longint)
8404:  procedure OleCheck(Result: HResult);
8405: end;
8406:
8407:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8408:  Function xGetActiveOleObject( const ClassName : string) : Variant
8409:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8410:  Function DllCanUnloadNow : HResult
8411:  Function DllRegisterServer : HResult
8412:  Function DllUnregisterServer : HResult
8413:  Function VarFromInterface( Unknown : IUnknown) : Variant
8414:  Function VarToInterface( const V : Variant) : IDispatch
8415:  Function VarToAutoObject( const V : Variant) : TAutoObject
8416:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8417:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8418:  Procedure OleError( ErrorCode : HResult)
8419:  Procedure OleCheck( Result : HResult)
8420:  Function StringToClassID( const S : string) : TCLSID
8421:  Function ClassIDToString( const ClassID : TCLSID) : string
8422:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8423:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8424:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8425:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8426:  Function xGUIDToString( const ClassID : TGUID) : string
8427:  Function xStringToGUID( const S : string) : TGUID
8428:  Function xGetModuleName( Module : HMODULE) : string
8429:  Function xAcquireExceptionObject : TObject
8430:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8431:  Function xUtf8Encode( const WS : WideString) : UTF8String
8432:  Function xUtf8Decode( const S : UTF8String) : WideString
8433:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8434:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8435:  Function XRTLHandleCOMException : HResult
8436:  Procedure XRTLCheckArgument( Flag : Boolean)
8437:  //Procedure XRTLCheckOutArgument( out Arg)
8438:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8439:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8440:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8441:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8442:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8443:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8444: function   XRTLDefaultCategoryManager: IUnknown;
8445: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8446: // ICatRegister helper functions
8447: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8448:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8449:                                        const CategoryManager: IUnknown = nil): HResult;
8450: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8451:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8452:                                        const CategoryManager: IUnknown = nil): HResult;
8453: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8454:                                        const CategoryManager: IUnknown = nil): HResult;
8455: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8456:                                          const CategoryManager: IUnknown = nil): HResult;
8457: // ICatInformation helper functions
8458: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8459:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8460:                                       const CategoryManager: IUnknown = nil): HResult;
8461: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8462:                                const CategoryManager: IUnknown = nil): HResult;
8463: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8464:                                     const CategoryManager: IUnknown = nil): HResult;
8465: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
```

```
8466:                                    const CategoryManager: IUnknown = nil): HResult;
8467: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8468:                    const ADelete: Boolean = True): WideString;
8469: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8470: Function XRTLGetVariantAsString( const Value : Variant) : string
8471: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8472: Function XRTLGetTimeZones : TXRTLTimeZones
8473: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8474: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8475: Function GMTNow : TDateTime
8476: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8477: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8478: Procedure XRTLNotImplemented
8479: Procedure XRTLRaiseError( E : Exception)
8480: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8481:
8482:
8483: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8484: begin
8485:   SIRegister_IXRTLValue(CL);
8486:   SIRegister_TXRTLValue(CL);
8487:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8488:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8489: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8490: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8491: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8492: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8493: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8494: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8495: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8496: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8497: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8498: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8499: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8500: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8501: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8502: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8503: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8504: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8505: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8506: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8507: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8508: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8509: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8510: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8511: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8512: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8513: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8514: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8515: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8516:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8517: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8518: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8519: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8520: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8521: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8522: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8523: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8524: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8525: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8526: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8527: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8528: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8529: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8530: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8531: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8532: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8533: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8534: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8535: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8536: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8537: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8538: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8539: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8540: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8541: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8542: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8543: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8544: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8545: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8546: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8547: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8548: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8549: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8550: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8551: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8552: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8553: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
```

```
8554: end;
8555:
8556: //*****************************unit uPSI_GR32;****************************************
8557:
8558:  Function Color32( WinColor : TColor) : TColor32;
8559:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8560:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8561:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8562:  Function WinColor( Color32 : TColor32) : TColor
8563:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8564:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8565:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8566:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8567:  Function RedComponent( Color32 : TColor32) : Integer
8568:  Function GreenComponent( Color32 : TColor32) : Integer
8569:  Function BlueComponent( Color32 : TColor32) : Integer
8570:  Function AlphaComponent( Color32 : TColor32) : Integer
8571:  Function Intensity( Color32 : TColor32) : Integer
8572:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8573:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8574:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8575:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8576:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8577:  Function WinPalette( const P : TPalette32) : HPALETTE
8578:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8579:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8580:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8581:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8582:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8583:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8584:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8585:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8586:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8587:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8588:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8589:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8590:  Function FixedRect1( const ARect : TRect) : TRect;
8591:  Function FixedRect2( const FR : TFloatRect) : TRect;
8592:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8593:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8594:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8595:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8596:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8597:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8598:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8599:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8600:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8601:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8602:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8603:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8604:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8605:  Function IsRectEmpty( const R : TRect) : Boolean;
8606:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8607:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8608:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8609:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8610:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8611:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8612:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8613:  Function MessageBeep( uType : UINT) : BOOL
8614:  Function ShowCursor( bShow : BOOL) : Integer
8615:  Function SetCursorPos( X, Y : Integer) : BOOL
8616:  Function SetCursor( hCursor : HICON) : HCURSOR
8617:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8618:  //Function ClipCursor( lpRect : PRect) : BOOL
8619:  Function GetClipCursor( var lpRect : TRect) : BOOL
8620:  Function GetCursor : HCURSOR
8621:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8622:  Function GetCaretBlinkTime : UINT
8623:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8624:  Function DestroyCaret : BOOL
8625:  Function HideCaret( hWnd : HWND) : BOOL
8626:  Function ShowCaret( hWnd : HWND) : BOOL
8627:  Function SetCaretPos( X, Y : Integer) : BOOL
8628:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8629:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8630:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8631:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8632:  Function WindowFromPoint( Point : TPoint) : HWND
8633:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8634:
8635:
8636: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8637: begin
8638:  Function FixedFloor( A : TFixed) : Integer
8639:  Function FixedCeil( A : TFixed) : Integer
8640:  Function FixedMul( A, B : TFixed) : TFixed
8641:  Function FixedDiv( A, B : TFixed) : TFixed
8642:  Function OneOver( Value : TFixed) : TFixed
```

```
8643:  Function FixedRound( A : TFixed) : Integer
8644:  Function FixedSqr( Value : TFixed) : TFixed
8645:  Function FixedSqrtLP( Value : TFixed) : TFixed
8646:  Function FixedSqrtHP( Value : TFixed) : TFixed
8647:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8648:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8649:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8650:  Function GRHypot( const X, Y : TFloat) : TFloat;
8651:  Function Hypot1( const X, Y : Integer) : Integer;
8652:  Function FastSqrt( const Value : TFloat) : TFloat
8653:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8654:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8655:  Function FastInvSqrt( const Value : Single) : Single;
8656:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8657:  Function GRIsPowerOf2( Value : Integer) : Boolean
8658:  Function PrevPowerOf2( Value : Integer) : Integer
8659:  Function NextPowerOf2( Value : Integer) : Integer
8660:  Function Average( A, B : Integer) : Integer
8661:  Function GRSign( Value : Integer) : Integer
8662:  Function FloatMod( x, y : Double) : Double
8663: end;
8664:
8665: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8666: begin
8667:  Function Clamp( const Value : Integer) : Integer;
8668:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8669:  Function StackAlloc( Size : Integer) : Pointer
8670:  Procedure StackFree( P : Pointer)
8671:  Procedure Swap( var A, B : Pointer);
8672:  Procedure Swap1( var A, B : Integer);
8673:  Procedure Swap2( var A, B : TFixed);
8674:  Procedure Swap3( var A, B : TColor32);
8675:  Procedure TestSwap( var A, B : Integer);
8676:  Procedure TestSwap1( var A, B : TFixed);
8677:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8678:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8679:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8680:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8681:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8682:  Function GRMin( const A, B, C : Integer) : Integer;
8683:  Function GRMax( const A, B, C : Integer) : Integer;
8684:  Function Clamp( Value, Max : Integer) : Integer;
8685:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8686:  Function Wrap( Value, Max : Integer) : Integer;
8687:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8688:  Function Wrap3( Value, Max : Single) : Single;;
8689:  Function WrapPow2( Value, Max : Integer) : Integer;
8690:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8691:  Function Mirror( Value, Max : Integer) : Integer;
8692:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8693:  Function MirrorPow2( Value, Max : Integer) : Integer;
8694:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8695:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8696:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8697:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8698:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8699:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8700:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8701:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8702:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8703:  Function Div255( Value : Cardinal) : Cardinal
8704:  Function SAR_4( Value : Integer) : Integer
8705:  Function SAR_8( Value : Integer) : Integer
8706:  Function SAR_9( Value : Integer) : Integer
8707:  Function SAR_11( Value : Integer) : Integer
8708:  Function SAR_12( Value : Integer) : Integer
8709:  Function SAR_13( Value : Integer) : Integer
8710:  Function SAR_14( Value : Integer) : Integer
8711:  Function SAR_15( Value : Integer) : Integer
8712:  Function SAR_16( Value : Integer) : Integer
8713:  Function ColorSwap( WinColor : TColor) : TColor32
8714: end;
8715:
8716: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8717: begin
8718:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8719:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8720:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8721:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8722:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8723:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8724:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8725:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8726:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8727:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8728:  Function CreateBitmask( Components : TColor32Components) : TColor32
8729:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
```

```
8730:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8731:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8732: end;
8733:
8734:
8735: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8736: begin
8737:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8738:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8739:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8740:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8741:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8742:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8743:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8744:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8745:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8746:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8747:    //+'tedRangeBuffer; MoreData : Boolean; end
8748:  Function NtfsSetSparse( const FileName : string) : Boolean
8749:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8750:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8751:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8752:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8753:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8754:  Function NtfsGetSparse( const FileName : string) : Boolean
8755:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8756:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8757:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8758:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8759:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8760:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8761:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8762:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8763:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8764:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8765:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8766:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8767:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8768:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8769:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8770:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8771:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8772:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8773:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8774:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8775:   AddTypeS('TStreamIds', 'set of TStreamId
8776:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8777:    +': ___Pointer; StreamIds : TStreamIds; end
8778:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8779:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8780:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8781:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8782:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8783:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8784:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8785:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8786:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8787:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8788:  Function JclAppInstances : TJclAppInstances;
8789:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8790:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8791:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8792:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8793:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8794:
8795:
8796: (*----------------------------------------------------------------------------*)
8797: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8798: begin
8799:   FindClass('TOBJECT'),'EJclGraphicsError
8800:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8801:   TDynPointArray', 'array of TPoint
8802:   TDynDynPointArrayArray', 'array of TDynPointArray
8803:   TPointF', 'record X : Single; Y : Single; end
8804:   TDynPointArrayF', 'array of TPointF
8805:   TDrawMode2', '( dmOpaque, dmBlend )
8806:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8807:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8808:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8809:   TMatrix3d', 'record array[0..2,0..2] of extended end
8810:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8811:   TScanLine', 'array of Integer
8812:   TScanLines', 'array of TScanLine
8813:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8814:   TGradientDirection', '( gdVertical, gdHorizontal )
```

```
8815:   TPolyFillMode', '( fmAlternate, fmWinding )
8816:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8817:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8818:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8819:   SIRegister_TJclDesktopCanvas(CL);
8820:   FindClass('TOBJECT'),'TJclRegion
8821:   SIRegister_TJclRegionInfo(CL);
8822:   SIRegister_TJclRegion(CL);
8823:   SIRegister_TJclThreadPersistent(CL);
8824:   SIRegister_TJclCustomMap(CL);
8825:   SIRegister_TJclBitmap32(CL);
8826:   SIRegister_TJclByteMap(CL);
8827:   SIRegister_TJclTransformation(CL);
8828:   SIRegister_TJclLinearTransformation(CL);
8829:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8830:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8831:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8832:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8833:  Procedure BitmapToJPeg( const FileName : string)
8834:  Procedure JPegToBitmap( const FileName : string)
8835:  Function ExtractIconCount( const FileName : string) : Integer
8836:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8837:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8838:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8839:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8840:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8841:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8842:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8843:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8844:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8845:  Procedure ScreenShot1( bm : TBitmap);
8846:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8847:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8848:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8849:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8850:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8851:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8852:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8853:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8854:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8855:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8856:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8857:  Procedure Invert( Dst, Src : TJclBitmap32)
8858:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8859:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8860:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8861:  Procedure SetGamma( Gamma : Single)
8862: end;
8863:
8864: (*----------------------------------------------------------------------------*)
8865: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8866: begin
8867:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8868:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8869:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8870:  Function LockedDec( var Target : Integer) : Integer
8871:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8872:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8873:  Function LockedExchangeDec( var Target : Integer) : Integer
8874:  Function LockedExchangeInc( var Target : Integer) : Integer
8875:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8876:  Function LockedInc( var Target : Integer) : Integer
8877:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8878:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8879:   SIRegister_TJclDispatcherObject(CL);
8880:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8881:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8882:   SIRegister_TJclCriticalSection(CL);
8883:   SIRegister_TJclCriticalSectionEx(CL);
8884:   SIRegister_TJclEvent(CL);
8885:   SIRegister_TJclWaitableTimer(CL);
8886:   SIRegister_TJclSemaphore(CL);
8887:   SIRegister_TJclMutex(CL);
8888:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8889:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8890:   SIRegister_TJclOptex(CL);
8891:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8892:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8893:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8894:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8895:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8896:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
```

```
8897:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8898:   PMeteredSection', '^TMeteredSection // will not work
8899:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8900:   SIRegister_TJclMeteredSection(CL);
8901:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8902:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8903:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8904:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8905:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8906:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8907:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8908:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8909:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8910:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8911:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8912:   FindClass('TOBJECT'),'EJclCriticalSectionError
8913:   FindClass('TOBJECT'),'EJclEventError
8914:   FindClass('TOBJECT'),'EJclWaitableTimerError
8915:   FindClass('TOBJECT'),'EJclSemaphoreError
8916:   FindClass('TOBJECT'),'EJclMutexError
8917:   FindClass('TOBJECT'),'EJclMeteredSectionError
8918: end;
8919:
8920:
8921: //*************************unit uPSI_mORMotReport;
8922: Procedure SetCurrentPrinterAsDefault
8923: Function CurrentPrinterName : string
8924: Function mCurrentPrinterPaperSize : string
8925: Procedure UseDefaultPrinter
8926:
8927: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
8928: begin
8929:   with FindClass('TOBJECT'), 'TStream') do begin
8930:     IsAbstract := True;
8931:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8932:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8933:     function Read(Buffer:String;Count:LongInt):LongInt
8934:     function Write(Buffer:String;Count:LongInt):LongInt
8935:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
8936:     function WriteString(Buffer:String;Count:LongInt):LongInt
8937:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8938:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
8939:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8940:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8941:
8942:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8943:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8944:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8945:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8946:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8947:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8948:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8949:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8950:
8951:     function Seek(Offset:LongInt;Origin:Word):LongInt
8952:     procedure ReadBuffer(Buffer:String;Count:LongInt)
8953:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8954:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
8955:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
8956:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
8957:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
8958:
8959:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8960:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8961:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8962:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8963:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8964:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8965:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8966:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8967:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8968:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8969:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
8970:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
8971:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
8972:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
8973:
8974:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
8975:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
8976:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
8977:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
8978:     //READBUFFERAC
8979:     function InstanceSize: Longint
8980:     Procedure FixupResourceHeader( FixupInfo : Integer)
8981:     Procedure ReadResHeader
8982:
8983:     {$IFDEF DELPHI4UP}
8984:     function CopyFrom(Source:TStream;Count:Int64):LongInt
8985:     {$ELSE}
```

```
8986:     function CopyFrom(Source:TStream;Count:Integer):LongInt
8987:     {$ENDIF}
8988:     RegisterProperty('Position', 'LongInt', iptrw);
8989:     RegisterProperty('Size', 'LongInt', iptrw);
8990:   end;
8991: end;
8992:
8993:
8994: { ****************************************************************
8995:   Unit DMATH - Interface for DMATH.DLL
8996:   **************************************************************** }
8997: // see more docs/dmath_manual.pdf
8998:
8999: Function InitEval : Integer
9000: Procedure SetVariable( VarName : Char; Value : Float)
9001: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9002: Function Eval( ExpressionString : String) : Float
9003:
9004: unit dmath; //types are in built, others are external in DLL
9005: interface
9006: {$IFDEF DELPHI}
9007: uses
9008:   StdCtrls, Graphics;
9009: {$ENDIF}
9010: { ----------------------------------------------------------------
9011:   Types and constants
9012:   ---------------------------------------------------------------- }
9013: {$i types.inc}
9014: { ----------------------------------------------------------------
9015:   Error handling
9016:   ---------------------------------------------------------------- }
9017: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9018: { Sets the error code }
9019: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9020: { Sets error code and default function value }
9021: function MathErr : Integer; external 'dmath';
9022: { Returns the error code }
9023: { ----------------------------------------------------------------
9024:   Dynamic arrays
9025:   ---------------------------------------------------------------- }
9026: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9027: { Sets the auto-initialization of arrays }
9028: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9029: { Creates floating point vector V[0..Ub] }
9030: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9031: { Creates integer vector V[0..Ub] }
9032: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9033: { Creates complex vector V[0..Ub] }
9034: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9035: { Creates boolean vector V[0..Ub] }
9036: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9037: { Creates string vector V[0..Ub] }
9038: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9039: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9040: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9041: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9042: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9043: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9044: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9045: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9046: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9047: { Creates string matrix A[0..Ub1, 0..Ub2] }
9048: { ----------------------------------------------------------------
9049:   Minimum, maximum, sign and exchange
9050:   ---------------------------------------------------------------- }
9051: function FMin(X, Y : Float) : Float; external 'dmath';
9052: { Minimum of 2 reals }
9053: function FMax(X, Y : Float) : Float; external 'dmath';
9054: { Maximum of 2 reals }
9055: function IMin(X, Y : Integer) : Integer; external 'dmath';
9056: { Minimum of 2 integers }
9057: function IMax(X, Y : Integer) : Integer; external 'dmath';
9058: { Maximum of 2 integers }
9059: function Sgn(X : Float) : Integer; external 'dmath';
9060: { Sign (returns 1 if X = 0) }
9061: function Sgn0(X : Float) : Integer; external 'dmath';
9062: { Sign (returns 0 if X = 0) }
9063: function DSgn(A, B : Float) : Float; external 'dmath';
9064: { Sgn(B) * |A| }
9065: procedure FSwap(var X, Y : Float); external 'dmath';
9066: { Exchange 2 reals }
9067: procedure ISwap(var X, Y : Integer); external 'dmath';
9068: { Exchange 2 integers }
9069: { ----------------------------------------------------------------
9070:   Rounding functions
9071:   ---------------------------------------------------------------- }
9072: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9073: { Rounds X to N decimal places }
9074: function Ceil(X : Float) : Integer; external 'dmath';
```

```
9075: { Ceiling function }
9076: function Floor(X : Float) : Integer; external 'dmath';
9077: { Floor function }
9078: { ------------------------------------------------------------
9079:   Logarithms, exponentials and power
9080:   ------------------------------------------------------------ }
9081: function Expo(X : Float) : Float; external 'dmath';
9082: { Exponential }
9083: function Exp2(X : Float) : Float; external 'dmath';
9084: { 2^X }
9085: function Exp10(X : Float) : Float; external 'dmath';
9086: { 10^X }
9087: function Log(X : Float) : Float; external 'dmath';
9088: { Natural log }
9089: function Log2(X : Float) : Float; external 'dmath';
9090: { Log, base 2 }
9091: function Log10(X : Float) : Float; external 'dmath';
9092: { Decimal log }
9093: function LogA(X, A : Float) : Float; external 'dmath';
9094: { Log, base A }
9095: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9096: { X^N }
9097: function Power(X, Y : Float) : Float; external 'dmath';
9098: { X^Y, X >= 0 }
9099: { ------------------------------------------------------------
9100:   Trigonometric functions
9101:   ------------------------------------------------------------ }
9102: function Pythag(X, Y : Float) : Float; external 'dmath';
9103: { Sqrt(X^2 + Y^2) }
9104: function FixAngle(Theta : Float) : Float; external 'dmath';
9105: { Set Theta in -Pi..Pi }
9106: function Tan(X : Float) : Float; external 'dmath';
9107: { Tangent }
9108: function ArcSin(X : Float) : Float; external 'dmath';
9109: { Arc sinus }
9110: function ArcCos(X : Float) : Float; external 'dmath';
9111: { Arc cosinus }
9112: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9113: { Angle (Ox, OM) with M(X,Y) }
9114: { ------------------------------------------------------------
9115:   Hyperbolic functions
9116:   ------------------------------------------------------------ }
9117: function Sinh(X : Float) : Float; external 'dmath';
9118: { Hyperbolic sine }
9119: function Cosh(X : Float) : Float; external 'dmath';
9120: { Hyperbolic cosine }
9121: function Tanh(X : Float) : Float; external 'dmath';
9122: { Hyperbolic tangent }
9123: function ArcSinh(X : Float) : Float; external 'dmath';
9124: { Inverse hyperbolic sine }
9125: function ArcCosh(X : Float) : Float; external 'dmath';
9126: { Inverse hyperbolic cosine }
9127: function ArcTanh(X : Float) : Float; external 'dmath';
9128: { Inverse hyperbolic tangent }
9129: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9130: { Sinh & Cosh }
9131: { ------------------------------------------------------------
9132:   Gamma function and related functions
9133:   ------------------------------------------------------------ }
9134: function Fact(N : Integer) : Float; external 'dmath';
9135: { Factorial }
9136: function SgnGamma(X : Float) : Integer; external 'dmath';
9137: { Sign of Gamma function }
9138: function Gamma(X : Float) : Float; external 'dmath';
9139: { Gamma function }
9140: function LnGamma(X : Float) : Float; external 'dmath';
9141: { Logarithm of Gamma function }
9142: function Stirling(X : Float) : Float; external 'dmath';
9143: { Stirling's formula for the Gamma function }
9144: function StirLog(X : Float) : Float; external 'dmath';
9145: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9146: function DiGamma(X : Float ) : Float; external 'dmath';
9147: { Digamma function }
9148: function TriGamma(X : Float ) : Float; external 'dmath';
9149: { Trigamma function }
9150: function IGamma(A, X : Float) : Float; external 'dmath';
9151: { Incomplete Gamma function}
9152: function JGamma(A, X : Float) : Float; external 'dmath';
9153: { Complement of incomplete Gamma function }
9154: function InvGamma(A, P : Float) : Float; external 'dmath';
9155: { Inverse of incomplete Gamma function }
9156: function Erf(X : Float) : Float; external 'dmath';
9157: { Error function }
9158: function Erfc(X : Float) : Float; external 'dmath';
9159: { Complement of error function }
9160: { ------------------------------------------------------------
9161:   Beta function and related functions
9162:   ------------------------------------------------------------ }
9163: function Beta(X, Y : Float) : Float; external 'dmath';
```

```
9164: { Beta function }
9165: function IBeta(A, B, X : Float) : Float; external 'dmath';
9166: { Incomplete Beta function }
9167: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9168: { Inverse of incomplete Beta function }
9169: { -----------------------------------------------------------------
9170:   Lambert's function
9171:   ----------------------------------------------------------------- }
9172: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9173:  -----------------------------------------------------------------
9174:   Binomial distribution
9175:   ----------------------------------------------------------------- }
9176: function Binomial(N, K : Integer) : Float; external 'dmath';
9177: { Binomial coefficient C(N,K) }
9178: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9179: { Probability of binomial distribution }
9180: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9181: { Cumulative probability for binomial distrib. }
9182: { -----------------------------------------------------------------
9183:   Poisson distribution
9184:   ----------------------------------------------------------------- }
9185: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9186: { Probability of Poisson distribution }
9187: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9188: { Cumulative probability for Poisson distrib. }
9189: { -----------------------------------------------------------------
9190:   Exponential distribution
9191:   ----------------------------------------------------------------- }
9192: function DExpo(A, X : Float) : Float; external 'dmath';
9193: { Density of exponential distribution with parameter A }
9194: function FExpo(A, X : Float) : Float; external 'dmath';
9195: { Cumulative probability function for exponential dist. with parameter A }
9196: { -----------------------------------------------------------------
9197:   Standard normal distribution
9198:   ----------------------------------------------------------------- }
9199: function DNorm(X : Float) : Float; external 'dmath';
9200: { Density of standard normal distribution }
9201: function FNorm(X : Float) : Float; external 'dmath';
9202: { Cumulative probability for standard normal distrib. }
9203: function PNorm(X : Float) : Float; external 'dmath';
9204: { Prob(|U| > X) for standard normal distrib. }
9205: function InvNorm(P : Float) : Float; external 'dmath';
9206: { Inverse of standard normal distribution }
9207: { -----------------------------------------------------------------
9208:   Student's distribution
9209:   ----------------------------------------------------------------- }
9210: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9211: { Density of Student distribution with Nu d.o.f. }
9212: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9213: { Cumulative probability for Student distrib. with Nu d.o.f. }
9214: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9215: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9216: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9217: { Inverse of Student's t-distribution function }
9218: { -----------------------------------------------------------------
9219:   Khi-2 distribution
9220:   ----------------------------------------------------------------- }
9221: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9222: { Density of Khi-2 distribution with Nu d.o.f. }
9223: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9224: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9225: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9226: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9227: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9228: { Inverse of Khi-2 distribution function }
9229: { -----------------------------------------------------------------
9230:   Fisher-Snedecor distribution
9231:   ----------------------------------------------------------------- }
9232: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9233: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9234: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9235: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9236: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9237: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9238: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9239: { Inverse of Snedecor's F-distribution function }
9240: { -----------------------------------------------------------------
9241:   Beta distribution
9242:   ----------------------------------------------------------------- }
9243: function DBeta(A, B, X : Float) : Float; external 'dmath';
9244: { Density of Beta distribution with parameters A and B }
9245: function FBeta(A, B, X : Float) : Float; external 'dmath';
9246: { Cumulative probability for Beta distrib. with param. A and B }
9247: { -----------------------------------------------------------------
9248:   Gamma distribution
9249:   ----------------------------------------------------------------- }
9250: function DGamma(A, B, X : Float) : Float; external 'dmath';
9251: { Density of Gamma distribution with parameters A and B }
9252: function FGamma(A, B, X : Float) : Float; external 'dmath';
```

```
9253: { Cumulative probability for Gamma distrib. with param. A and B }
9254: { ------------------------------------------------------------
9255:   Expression evaluation
9256:   ------------------------------------------------------------ }
9257: function InitEval : Integer; external 'dmath';
9258: { Initializes built-in functions and returns their number }
9259: function Eval(ExpressionString : String) : Float; external 'dmath';
9260: { Evaluates an expression at run-time }
9261: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9262: { Assigns a value to a variable }
9263: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9264: { Adds a function to the parser }
9265: { ------------------------------------------------------------
9266:   Matrices and linear equations
9267:   ------------------------------------------------------------ }
9268: procedure GaussJordan(A           : TMatrix;
9269:                       Lb, Ub1, Ub2 : Integer;
9270:                       var Det      : Float); external 'dmath';
9271: { Transforms a matrix according to the Gauss-Jordan method }
9272: procedure LinEq(A       : TMatrix;
9273:                 B       : TVector;
9274:                 Lb, Ub  : Integer;
9275:                 var Det : Float); external 'dmath';
9276: { Solves a linear system according to the Gauss-Jordan method }
9277: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9278: { Cholesky factorization of a positive definite symmetric matrix }
9279: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9280: { LU decomposition }
9281: procedure LU_Solve(A      : TMatrix;
9282:                    B      : TVector;
9283:                    Lb, Ub : Integer;
9284:                    X      : TVector); external 'dmath';
9285: { Solution of linear system from LU decomposition }
9286: procedure QR_Decomp(A           : TMatrix;
9287:                     Lb, Ub1, Ub2 : Integer;
9288:                     R           : TMatrix); external 'dmath';
9289: { QR decomposition }
9290: procedure QR_Solve(Q, R         : TMatrix;
9291:                    B           : TVector;
9292:                    Lb, Ub1, Ub2 : Integer;
9293:                    X           : TVector); external 'dmath';
9294: { Solution of linear system from QR decomposition }
9295: procedure SV_Decomp(A           : TMatrix;
9296:                     Lb, Ub1, Ub2 : Integer;
9297:                     S           : TVector;
9298:                     V           : TMatrix); external 'dmath';
9299: { Singular value decomposition }
9300: procedure SV_SetZero(S      : TVector;
9301:                      Lb, Ub : Integer;
9302:                      Tol    : Float); external 'dmath';
9303: { Set lowest singular values to zero }
9304: procedure SV_Solve(U           : TMatrix;
9305:                    S           : TVector;
9306:                    V           : TMatrix;
9307:                    B           : TVector;
9308:                    Lb, Ub1, Ub2 : Integer;
9309:                    X           : TVector); external 'dmath';
9310: { Solution of linear system from SVD }
9311: procedure SV_Approx(U           : TMatrix;
9312:                     S           : TVector;
9313:                     V           : TMatrix;
9314:                     Lb, Ub1, Ub2 : Integer;
9315:                     A           : TMatrix); external 'dmath';
9316: { Matrix approximation from SVD }
9317: procedure EigenVals(A      : TMatrix;
9318:                     Lb, Ub : Integer;
9319:                     Lambda : TCompVector); external 'dmath';
9320: { Eigenvalues of a general square matrix }
9321: procedure EigenVect(A      : TMatrix;
9322:                     Lb, Ub : Integer;
9323:                     Lambda : TCompVector;
9324:                     V      : TMatrix); external 'dmath';
9325: { Eigenvalues and eigenvectors of a general square matrix }
9326: procedure EigenSym(A      : TMatrix;
9327:                    Lb, Ub : Integer;
9328:                    Lambda : TVector;
9329:                    V      : TMatrix); external 'dmath';
9330: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9331: procedure Jacobi(A              : TMatrix;
9332:                  Lb, Ub, MaxIter : Integer;
9333:                  Tol            : Float;
9334:                  Lambda         : TVector;
9335:                  V              : TMatrix); external 'dmath';
9336: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9337: { ------------------------------------------------------------
9338:   Optimization
9339:   ------------------------------------------------------------ }
9340: procedure MinBrack(Func                    : TFunc;
9341:                    var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
```

```
9342: { Brackets a minimum of a function }
9343: procedure GoldSearch(Func           : TFunc;
9344:                      A, B           : Float;
9345:                      MaxIter        : Integer;
9346:                      Tol            : Float;
9347:                      var Xmin, Ymin : Float); external 'dmath';
9348: { Minimization of a function of one variable (golden search) }
9349: procedure LinMin(Func      : TFuncNVar;
9350:                  X, DeltaX : TVector;
9351:                  Lb, Ub    : Integer;
9352:                  var R     : Float;
9353:                  MaxIter   : Integer;
9354:                  Tol       : Float;
9355:                  var F_min : Float); external 'dmath';
9356: { Minimization of a function of several variables along a line }
9357: procedure Newton(Func      : TFuncNVar;
9358:                  HessGrad  : THessGrad;
9359:                  X         : TVector;
9360:                  Lb, Ub    : Integer;
9361:                  MaxIter   : Integer;
9362:                  Tol       : Float;
9363:                  var F_min : Float;
9364:                  G         : TVector;
9365:                  H_inv     : TMatrix;
9366:                  var Det   : Float); external 'dmath';
9367: { Minimization of a function of several variables (Newton's method) }
9368: procedure SaveNewton(FileName : string); external 'dmath';
9369: { Save Newton iterations in a file }
9370: procedure Marquardt(Func      : TFuncNVar;
9371:                     HessGrad  : THessGrad;
9372:                     X         : TVector;
9373:                     Lb, Ub    : Integer;
9374:                     MaxIter   : Integer;
9375:                     Tol       : Float;
9376:                     var F_min : Float;
9377:                     G         : TVector;
9378:                     H_inv     : TMatrix;
9379:                     var Det   : Float); external 'dmath';
9380: { Minimization of a function of several variables (Marquardt's method) }
9381: procedure SaveMarquardt(FileName : string); external 'dmath';
9382: { Save Marquardt iterations in a file }
9383: procedure BFGS(Func      : TFuncNVar;
9384:                Gradient  : TGradient;
9385:                X         : TVector;
9386:                Lb, Ub    : Integer;
9387:                MaxIter   : Integer;
9388:                Tol       : Float;
9389:                var F_min : Float;
9390:                G         : TVector;
9391:                H_inv     : TMatrix); external 'dmath';
9392: { Minimization of a function of several variables (BFGS method) }
9393: procedure SaveBFGS(FileName : string); external 'dmath';
9394: { Save BFGS iterations in a file }
9395: procedure Simplex(Func      : TFuncNVar;
9396:                   X         : TVector;
9397:                   Lb, Ub    : Integer;
9398:                   MaxIter   : Integer;
9399:                   Tol       : Float;
9400:                   var F_min : Float); external 'dmath';
9401: { Minimization of a function of several variables (Simplex) }
9402: procedure SaveSimplex(FileName : string); external 'dmath';
9403: { Save Simplex iterations in a file }
9404: { ---------------------------------------------------------------
9405:   Nonlinear equations
9406:   --------------------------------------------------------------- }
9407: procedure RootBrack(Func             : TFunc;
9408:                     var X, Y, FX, FY : Float); external 'dmath';
9409: { Brackets a root of function Func between X and Y }
9410: procedure Bisect(Func     : TFunc;
9411:                  var X, Y : Float;
9412:                  MaxIter  : Integer;
9413:                  Tol      : Float;
9414:                  var F    : Float); external 'dmath';
9415: { Bisection method }
9416: procedure Secant(Func     : TFunc;
9417:                  var X, Y : Float;
9418:                  MaxIter  : Integer;
9419:                  Tol      : Float;
9420:                  var F    : Float); external 'dmath';
9421: { Secant method }
9422: procedure NewtEq(Func, Deriv : TFunc;
9423:                  var X       : Float;
9424:                  MaxIter     : Integer;
9425:                  Tol         : Float;
9426:                  var F       : Float); external 'dmath';
9427: { Newton-Raphson method for a single nonlinear equation }
9428: procedure NewtEqs(Equations : TEquations;
9429:                   Jacobian  : TJacobian;
9430:                   X, F      : TVector;
```

```
9431:                   Lb, Ub    : Integer;
9432:                   MaxIter   : Integer;
9433:                   Tol       : Float); external 'dmath';
9434: { Newton-Raphson method for a system of nonlinear equations }
9435: procedure Broyden(Equations : TEquations;
9436:                   X, F      : TVector;
9437:                   Lb, Ub    : Integer;
9438:                   MaxIter   : Integer;
9439:                   Tol       : Float); external 'dmath';
9440: { Broyden's method for a system of nonlinear equations }
9441: { ---------------------------------------------------------------
9442:   Polynomials and rational fractions
9443:   --------------------------------------------------------------- }
9444: function Poly(X    : Float;
9445:               Coef : TVector;
9446:               Deg  : Integer) : Float; external 'dmath';
9447: { Evaluates a polynomial }
9448: function RFrac(X          : Float;
9449:                Coef       : TVector;
9450:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9451: { Evaluates a rational fraction }
9452: function RootPol1(A, B  : Float;
9453:                   var X : Float) : Integer; external 'dmath';
9454: { Solves the linear equation A + B * X = 0 }
9455: function RootPol2(Coef : TVector;
9456:                   Z    : TCompVector) : Integer; external 'dmath';
9457: { Solves a quadratic equation }
9458: function RootPol3(Coef : TVector;
9459:                   Z    : TCompVector) : Integer; external 'dmath';
9460: { Solves a cubic equation }
9461: function RootPol4(Coef : TVector;
9462:                   Z    : TCompVector) : Integer; external 'dmath';
9463: { Solves a quartic equation }
9464: function RootPol(Coef : TVector;
9465:                  Deg  : Integer;
9466:                  Z    : TCompVector) : Integer; external 'dmath';
9467: { Solves a polynomial equation }
9468: function SetRealRoots(Deg : Integer;
9469:                       Z   : TCompVector;
9470:                       Tol : Float) : Integer; external 'dmath';
9471: { Set the imaginary part of a root to zero }
9472: procedure SortRoots(Deg : Integer;
9473:                     Z   : TCompVector); external 'dmath';
9474: { Sorts the roots of a polynomial }
9475: { ---------------------------------------------------------------
9476:   Numerical integration and differential equations
9477:   --------------------------------------------------------------- }
9478: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9479: { Integration by trapezoidal rule }
9480: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9481: { Integral from A to B }
9482: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9483: { Integral from 0 to B }
9484: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9485: { Convolution product at time T }
9486: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9487: { Convolution by trapezoidal rule }
9488: procedure RKF45(F                   : TDiffEqs;
9489:                 Neqn                : Integer;
9490:                 Y, Yp               : TVector;
9491:                 var T               : Float;
9492:                 Tout, RelErr, AbsErr : Float;
9493:                 var Flag            : Integer); external 'dmath';
9494: { Integration of a system of differential equations }
9495: { ---------------------------------------------------------------
9496:   Fast Fourier Transform
9497:   --------------------------------------------------------------- }
9498: procedure FFT(NumSamples        : Integer;
9499:               InArray, OutArray : TCompVector); external 'dmath';
9500: { Fast Fourier Transform }
9501: procedure IFFT(NumSamples        : Integer;
9502:                InArray, OutArray : TCompVector); external 'dmath';
9503: { Inverse Fast Fourier Transform }
9504: procedure FFT_Integer(NumSamples     : Integer;
9505:                       RealIn, ImagIn : TIntVector;
9506:                       OutArray       : TCompVector); external 'dmath';
9507: { Fast Fourier Transform for integer data }
9508: procedure FFT_Integer_Cleanup; external 'dmath';
9509: { Clear memory after a call to FFT_Integer }
9510: procedure CalcFrequency(NumSamples,
9511:                         FrequencyIndex : Integer;
9512:                         InArray        : TCompVector;
9513:                         var FFT        : Complex); external 'dmath';
9514: { Direct computation of Fourier transform }
9515: { ---------------------------------------------------------------
9516:   Random numbers
9517:   --------------------------------------------------------------- }
9518: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9519: { Select generator }
```

```
9520: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9521: { Initialize generator }
9522: function IRanGen : RNG_IntType; external 'dmath';
9523: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9524: function IRanGen31 : RNG_IntType; external 'dmath';
9525: { 31-bit random integer in [0 .. 2^31 - 1] }
9526: function RanGen1 : Float; external 'dmath';
9527: { 32-bit random real in [0,1] }
9528: function RanGen2 : Float; external 'dmath';
9529: { 32-bit random real in [0,1) }
9530: function RanGen3 : Float; external 'dmath';
9531: { 32-bit random real in (0,1) }
9532: function RanGen53 : Float; external 'dmath';
9533: { 53-bit random real in [0,1) }
9534: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9535: { Initializes the 'Multiply with carry' random number generator }
9536: function IRanMWC : RNG_IntType; external 'dmath';
9537: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9538: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9539: { Initializes Mersenne Twister generator with a seed }
9540: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9541:                         KeyLength : Word); external 'dmath';
9542: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9543: function IRanMT : RNG_IntType; external 'dmath';
9544: { Random integer from MT generator }
9545: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9546: { Initializes the UVAG generator with a string }
9547: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9548: { Initializes the UVAG generator with an integer }
9549: function IRanUVAG : RNG_IntType; external 'dmath';
9550: { Random integer from UVAG generator }
9551: function RanGaussStd : Float; external 'dmath';
9552: { Random number from standard normal distribution }
9553: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9554: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9555: procedure RanMult(M      : TVector; L      : TMatrix;
9556:                   Lb, Ub : Integer;
9557:                   X      : TVector); external 'dmath';
9558: { Random vector from multinormal distribution (correlated) }
9559: procedure RanMultIndep(M, S   : TVector;
9560:                        Lb, Ub : Integer;
9561:                        X      : TVector); external 'dmath';
9562: { Random vector from multinormal distribution (uncorrelated) }
9563: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9564: { Initializes Metropolis-Hastings parameters }
9565: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9566: { Returns Metropolis-Hastings parameters }
9567: procedure Hastings(Func      : TFuncNVar;
9568:                    T         : Float;
9569:                    X         : TVector;
9570:                    V         : TMatrix;
9571:                    Lb, Ub    : Integer;
9572:                    Xmat      : TMatrix;
9573:                    X_min     : TVector;
9574:                    var F_min : Float); external 'dmath';
9575: { Simulation of a probability density function by Metropolis-Hastings }
9576: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9577: { Initializes Simulated Annealing parameters }
9578: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9579: { Initializes log file }
9580: procedure SimAnn(Func          : TFuncNVar;
9581:                  X, Xmin, Xmax : TVector;
9582:                  Lb, Ub        : Integer;
9583:                  var F_min     : Float); external 'dmath';
9584: { Minimization of a function of several var. by simulated annealing }
9585: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9586: { Initializes Genetic Algorithm parameters }
9587: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9588: { Initializes log file }
9589: procedure GenAlg(Func          : TFuncNVar;
9590:                  X, Xmin, Xmax : TVector;
9591:                  Lb, Ub        : Integer;
9592:                  var F_min     : Float); external 'dmath';
9593: { Minimization of a function of several var. by genetic algorithm }
9594: { ------------------------------------------------------------------
9595:   Statistics
9596:   ------------------------------------------------------------------ }
9597: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9598: { Mean of sample X }
9599: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9600: { Minimum of sample X }
9601: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9602: { Maximum of sample X }
9603: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9604: { Median of sample X }
9605: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9606: { Standard deviation estimated from sample X }
9607: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9608: { Standard deviation of population }
```

```
9609: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9610: { Correlation coefficient }
9611: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9612: { Skewness of sample X }
9613: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9614: { Kurtosis of sample X }
9615: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9616: { Quick sort (ascending order) }
9617: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9618: { Quick sort (descending order) }
9619: procedure Interval(X1, X2              : Float;
9620:                    MinDiv, MaxDiv      : Integer;
9621:                    var Min, Max, Step : Float); external 'dmath';
9622: { Determines an interval for a set of values }
9623: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9624:                     var XMin, XMax, XStep : Float); external 'dmath';
9625: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9626: procedure StudIndep(N1, N2         : Integer;
9627:                     M1, M2, S1, S2 : Float;
9628:                     var T          : Float;
9629:                     var DoF        : Integer); external 'dmath';
9630: { Student t-test for independent samples }
9631: procedure StudPaired(X, Y    : TVector;
9632:                      Lb, Ub  : Integer;
9633:                      var T   : Float;
9634:                      var DoF : Integer); external 'dmath';
9635: { Student t-test for paired samples }
9636: procedure AnOVa1(Ns               : Integer;
9637:                  N                : TIntVector;
9638:                  M, S             : TVector;
9639:                  var V_f, V_r, F  : Float;
9640:                  var DoF_f, DoF_r : Integer); external 'dmath';
9641: { One-way analysis of variance }
9642: procedure AnOVa2(NA, NB, Nobs : Integer;
9643:                  M, S         : TMatrix;
9644:                  V, F         : TVector;
9645:                  DoF          : TIntVector); external 'dmath';
9646: { Two-way analysis of variance }
9647: procedure Snedecor(N1, N2         : Integer;
9648:                    S1, S2         : Float;
9649:                    var F          : Float;
9650:                    var DoF1, DoF2 : Integer); external 'dmath';
9651: { Snedecor's F-test (comparison of two variances) }
9652: procedure Bartlett(Ns       : Integer;
9653:                    N        : TIntVector;
9654:                    S        : TVector;
9655:                    var Khi2 : Float;
9656:                    var DoF  : Integer); external 'dmath';
9657: { Bartlett's test (comparison of several variances) }
9658: procedure Mann_Whitney(N1, N2     : Integer;
9659:                        X1, X2     : TVector;
9660:                        var U, Eps : Float); external 'dmath';
9661: { Mann-Whitney test}
9662: procedure Wilcoxon(X, Y       : TVector;
9663:                    Lb, Ub     : Integer;
9664:                    var Ndiff  : Integer;
9665:                    var T, Eps : Float); external 'dmath';
9666: { Wilcoxon test }
9667: procedure Kruskal_Wallis(Ns      : Integer;
9668:                          N       : TIntVector;
9669:                          X       : TMatrix;
9670:                          var H   : Float;
9671:                          var DoF : Integer); external 'dmath';
9672: { Kruskal-Wallis test }
9673: procedure Khi2_Conform(N_cls    : Integer;
9674:                        N_estim  : Integer;
9675:                        Obs      : TIntVector;
9676:                        Calc     : TVector;
9677:                        var Khi2 : Float;
9678:                        var DoF  : Integer); external 'dmath';
9679: { Khi-2 test for conformity }
9680: procedure Khi2_Indep(N_lin    : Integer;
9681:                      N_col    : Integer;
9682:                      Obs      : TIntMatrix;
9683:                      var Khi2 : Float;
9684:                      var DoF  : Integer); external 'dmath';
9685: { Khi-2 test for independence }
9686: procedure Woolf_Conform(N_cls   : Integer;
9687:                         N_estim : Integer;
9688:                         Obs     : TIntVector;
9689:                         Calc    : TVector;
9690:                         var G   : Float;
9691:                         var DoF : Integer); external 'dmath';
9692: { Woolf's test for conformity }
9693: procedure Woolf_Indep(N_lin   : Integer;
9694:                       N_col   : Integer;
9695:                       Obs     : TIntMatrix;
9696:                       var G   : Float;
9697:                       var DoF : Integer); external 'dmath';
```

```
9698: { Woolf's test for independence }
9699: procedure DimStatClassVector(var C : TStatClassVector;
9700:                              Ub    : Integer); external 'dmath';
9701: { Allocates an array of statistical classes: C[0..Ub] }
9702: procedure Distrib(X       : TVector;
9703:                   Lb, Ub  : Integer;
9704:                   A, B, H : Float;
9705:                   C       : TStatClassVector); external 'dmath';
9706: { Distributes an array X[Lb..Ub] into statistical classes }
9707: { ---------------------------------------------------------------
9708:   Linear / polynomial regression
9709:   --------------------------------------------------------------- }
9710: procedure LinFit(X, Y   : TVector;
9711:                  Lb, Ub : Integer;
9712:                  B      : TVector;
9713:                  V      : TMatrix); external 'dmath';
9714: { Linear regression : Y = B(0) + B(1) * X }
9715: procedure WLinFit(X, Y, S : TVector;
9716:                   Lb, Ub  : Integer;
9717:                   B       : TVector;
9718:                   V       : TMatrix); external 'dmath';
9719: { Weighted linear regression : Y = B(0) + B(1) * X }
9720: procedure SVDLinFit(X, Y   : TVector;
9721:                     Lb, Ub : Integer;
9722:                     SVDTol : Float;
9723:                     B      : TVector;
9724:                     V      : TMatrix); external 'dmath';
9725: { Unweighted linear regression by singular value decomposition }
9726: procedure WSVDLinFit(X, Y, S : TVector;
9727:                      Lb, Ub  : Integer;
9728:                      SVDTol  : Float;
9729:                      B       : TVector;
9730:                      V       : TMatrix); external 'dmath';
9731: { Weighted linear regression by singular value decomposition }
9732: procedure MulFit(X            : TMatrix;
9733:                  Y            : TVector;
9734:                  Lb, Ub, Nvar : Integer;
9735:                  ConsTerm     : Boolean;
9736:                  B            : TVector;
9737:                  V            : TMatrix); external 'dmath';
9738: { Multiple linear regression by Gauss-Jordan method }
9739: procedure WMulFit(X            : TMatrix;
9740:                   Y, S         : TVector;
9741:                   Lb, Ub, Nvar : Integer;
9742:                   ConsTerm     : Boolean;
9743:                   B            : TVector;
9744:                   V            : TMatrix); external 'dmath';
9745: { Weighted multiple linear regression by Gauss-Jordan method }
9746: procedure SVDFit(X            : TMatrix;
9747:                  Y            : TVector;
9748:                  Lb, Ub, Nvar : Integer;
9749:                  ConsTerm     : Boolean;
9750:                  SVDTol       : Float;
9751:                  B            : TVector;
9752:                  V            : TMatrix); external 'dmath';
9753: { Multiple linear regression by singular value decomposition }
9754: procedure WSVDFit(X            : TMatrix;
9755:                   Y, S         : TVector;
9756:                   Lb, Ub, Nvar : Integer;
9757:                   ConsTerm     : Boolean;
9758:                   SVDTol       : Float;
9759:                   B            : TVector;
9760:                   V            : TMatrix); external 'dmath';
9761: { Weighted multiple linear regression by singular value decomposition }
9762: procedure PolFit(X, Y        : TVector;
9763:                  Lb, Ub, Deg : Integer;
9764:                  B           : TVector;
9765:                  V           : TMatrix); external 'dmath';
9766: { Polynomial regression by Gauss-Jordan method }
9767: procedure WPolFit(X, Y, S     : TVector;
9768:                   Lb, Ub, Deg : Integer;
9769:                   B           : TVector;
9770:                   V           : TMatrix); external 'dmath';
9771: { Weighted polynomial regression by Gauss-Jordan method }
9772: procedure SVDPolFit(X, Y        : TVector;
9773:                     Lb, Ub, Deg : Integer;
9774:                     SVDTol      : Float;
9775:                     B           : TVector;
9776:                     V           : TMatrix); external 'dmath';
9777: { Unweighted polynomial regression by singular value decomposition }
9778: procedure WSVDPolFit(X, Y, S     : TVector;
9779:                      Lb, Ub, Deg : Integer;
9780:                      SVDTol      : Float;
9781:                      B           : TVector;
9782:                      V           : TMatrix); external 'dmath';
9783: { Weighted polynomial regression by singular value decomposition }
9784: procedure RegTest(Y, Ycalc : TVector;
9785:                   LbY, UbY : Integer;
9786:                   V        : TMatrix;
```

```
9787:                  LbV, UbV : Integer;
9788:                  var Test : TRegTest); external 'dmath';
9789: { Test of unweighted regression }
9790: procedure WRegTest(Y, Ycalc, S : TVector;
9791:                    LbY, UbY    : Integer;
9792:                    V           : TMatrix;
9793:                    LbV, UbV    : Integer;
9794:                    var Test    : TRegTest); external 'dmath';
9795: { Test of weighted regression }
9796: { ---------------------------------------------------------------
9797:   Nonlinear regression
9798:   --------------------------------------------------------------- }
9799: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9800: { Sets the optimization algorithm for nonlinear regression }
9801: function GetOptAlgo : TOptAlgo; external 'dmath';
9802: { Returns the optimization algorithm }
9803: procedure SetMaxParam(N : Byte); external 'dmath';
9804: { Sets the maximum number of regression parameters for nonlinear regression }
9805: function GetMaxParam : Byte; external 'dmath';
9806: { Returns the maximum number of regression parameters for nonlinear regression }
9807: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9808: { Sets the bounds on the I-th regression parameter }
9809: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9810: { Returns the bounds on the I-th regression parameter }
9811: procedure NLFit(RegFunc   : TRegFunc;
9812:                 DerivProc : TDerivProc;
9813:                 X, Y      : TVector;
9814:                 Lb, Ub    : Integer;
9815:                 MaxIter   : Integer;
9816:                 Tol       : Float;
9817:                 B         : TVector;
9818:                 FirstPar,
9819:                 LastPar   : Integer;
9820:                 V         : TMatrix); external 'dmath';
9821: { Unweighted nonlinear regression }
9822: procedure WNLFit(RegFunc   : TRegFunc;
9823:                  DerivProc : TDerivProc;
9824:                  X, Y, S   : TVector;
9825:                  Lb, Ub    : Integer;
9826:                  MaxIter   : Integer;
9827:                  Tol       : Float;
9828:                  B         : TVector;
9829:                  FirstPar,
9830:                  LastPar   : Integer;
9831:                  V         : TMatrix); external 'dmath';
9832: { Weighted nonlinear regression }
9833: procedure SetMCFile(FileName : String); external 'dmath';
9834: { Set file for saving MCMC simulations }
9835: procedure SimFit(RegFunc   : TRegFunc;
9836:                  X, Y      : TVector;
9837:                  Lb, Ub    : Integer;
9838:                  B         : TVector;
9839:                  FirstPar,
9840:                  LastPar   : Integer;
9841:                  V         : TMatrix); external 'dmath';
9842: { Simulation of unweighted nonlinear regression by MCMC }
9843: procedure WSimFit(RegFunc   : TRegFunc;
9844:                   X, Y, S   : TVector;
9845:                   Lb, Ub    : Integer;
9846:                   B         : TVector;
9847:                   FirstPar,
9848:                   LastPar   : Integer;
9849:                   V         : TMatrix); external 'dmath';
9850: { Simulation of weighted nonlinear regression by MCMC }
9851: { ---------------------------------------------------------------
9852:   Nonlinear regression models
9853:   --------------------------------------------------------------- }
9854: procedure FracFit(X, Y       : TVector;
9855:                   Lb, Ub     : Integer;
9856:                   Deg1, Deg2 : Integer;
9857:                   ConsTerm   : Boolean;
9858:                   MaxIter    : Integer;
9859:                   Tol        : Float;
9860:                   B          : TVector;
9861:                   V          : TMatrix); external 'dmath';
9862: { Unweighted fit of rational fraction }
9863: procedure WFracFit(X, Y, S    : TVector;
9864:                    Lb, Ub     : Integer;
9865:                    Deg1, Deg2 : Integer;
9866:                    ConsTerm   : Boolean;
9867:                    MaxIter    : Integer;
9868:                    Tol        : Float;
9869:                    B          : TVector;
9870:                    V          : TMatrix); external 'dmath';
9871: { Weighted fit of rational fraction }
9872:
9873: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9874: { Returns the value of the rational fraction at point X }
9875: procedure ExpFit(X, Y         : TVector;
```

```
9876:                 Lb, Ub, Nexp : Integer;
9877:                 ConsTerm     : Boolean;
9878:                 MaxIter      : Integer;
9879:                 Tol          : Float;
9880:                 B            : TVector;
9881:                 V            : TMatrix); external 'dmath';
9882: { Unweighted fit of sum of exponentials }
9883: procedure WExpFit(X, Y, S      : TVector;
9884:                  Lb, Ub, Nexp : Integer;
9885:                  ConsTerm     : Boolean;
9886:                  MaxIter      : Integer;
9887:                  Tol          : Float;
9888:                  B            : TVector;
9889:                  V            : TMatrix); external 'dmath';
9890: { Weighted fit of sum of exponentials }
9891: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9892: { Returns the value of the regression function at point X }
9893: procedure IncExpFit(X, Y     : TVector;
9894:                     Lb, Ub   : Integer;
9895:                     ConsTerm : Boolean;
9896:                     MaxIter  : Integer;
9897:                     Tol      : Float;
9898:                     B        : TVector;
9899:                     V        : TMatrix); external 'dmath';
9900: { Unweighted fit of model of increasing exponential }
9901: procedure WIncExpFit(X, Y, S  : TVector;
9902:                      Lb, Ub   : Integer;
9903:                      ConsTerm : Boolean;
9904:                      MaxIter  : Integer;
9905:                      Tol      : Float;
9906:                      B        : TVector;
9907:                      V        : TMatrix); external 'dmath';
9908: { Weighted fit of increasing exponential }
9909: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9910: { Returns the value of the regression function at point X }
9911: procedure ExpLinFit(X, Y    : TVector;
9912:                     Lb, Ub  : Integer;
9913:                     MaxIter : Integer;
9914:                     Tol     : Float;
9915:                     B       : TVector;
9916:                     V       : TMatrix); external 'dmath';
9917: { Unweighted fit of the "exponential + linear" model }
9918: procedure WExpLinFit(X, Y, S : TVector;
9919:                      Lb, Ub  : Integer;
9920:                      MaxIter : Integer;
9921:                      Tol     : Float;
9922:                      B       : TVector;
9923:                      V       : TMatrix); external 'dmath';
9924: { Weighted fit of the "exponential + linear" model }
9925:
9926: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9927: { Returns the value of the regression function at point X }
9928: procedure MichFit(X, Y    : TVector;
9929:                   Lb, Ub  : Integer;
9930:                   MaxIter : Integer;
9931:                   Tol     : Float;
9932:                   B       : TVector;
9933:                   V       : TMatrix); external 'dmath';
9934: { Unweighted fit of Michaelis equation }
9935: procedure WMichFit(X, Y, S : TVector;
9936:                    Lb, Ub  : Integer;
9937:                    MaxIter : Integer;
9938:                    Tol     : Float;
9939:                    B       : TVector;
9940:                    V       : TMatrix); external 'dmath';
9941: { Weighted fit of Michaelis equation }
9942: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9943: { Returns the value of the Michaelis equation at point X }
9944: procedure MintFit(X, Y    : TVector;
9945:                   Lb, Ub  : Integer;
9946:                   MintVar : TMintVar;
9947:                   Fit_S0  : Boolean;
9948:                   MaxIter : Integer;
9949:                   Tol     : Float;
9950:                   B       : TVector;
9951:                   V       : TMatrix); external 'dmath';
9952: { Unweighted fit of the integrated Michaelis equation }
9953: procedure WMintFit(X, Y, S : TVector;
9954:                    Lb, Ub  : Integer;
9955:                    MintVar : TMintVar;
9956:                    Fit_S0  : Boolean;
9957:                    MaxIter : Integer;
9958:                    Tol     : Float;
9959:                    B       : TVector;
9960:                    V       : TMatrix); external 'dmath';
9961: { Weighted fit of the integrated Michaelis equation }
9962: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9963: { Returns the value of the integrated Michaelis equation at point X }
9964: procedure HillFit(X, Y    : TVector;
```

```
 9965:                   Lb, Ub  : Integer;
 9966:                   MaxIter : Integer;
 9967:                   Tol     : Float;
 9968:                   B       : TVector;
 9969:                   V       : TMatrix); external 'dmath';
 9970: { Unweighted fit of Hill equation }
 9971: procedure WHillFit(X, Y, S : TVector;
 9972:                    Lb, Ub  : Integer;
 9973:                    MaxIter : Integer;
 9974:                    Tol     : Float;
 9975:                    B       : TVector;
 9976:                    V       : TMatrix); external 'dmath';
 9977: { Weighted fit of Hill equation }
 9978: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9979: { Returns the value of the Hill equation at point X }
 9980: procedure LogiFit(X, Y     : TVector;
 9981:                   Lb, Ub   : Integer;
 9982:                   ConsTerm : Boolean;
 9983:                   General  : Boolean;
 9984:                   MaxIter  : Integer;
 9985:                   Tol      : Float;
 9986:                   B        : TVector;
 9987:                   V        : TMatrix); external 'dmath';
 9988: { Unweighted fit of logistic function }
 9989: procedure WLogiFit(X, Y, S  : TVector;
 9990:                    Lb, Ub   : Integer;
 9991:                    ConsTerm : Boolean;
 9992:                    General  : Boolean;
 9993:                    MaxIter  : Integer;
 9994:                    Tol      : Float;
 9995:                    B        : TVector;
 9996:                    V        : TMatrix); external 'dmath';
 9997: { Weighted fit of logistic function }
 9998: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9999: { Returns the value of the logistic function at point X }
10000: procedure PKFit(X, Y    : TVector;
10001:                 Lb, Ub  : Integer;
10002:                 MaxIter : Integer;
10003:                 Tol     : Float;
10004:                 B       : TVector;
10005:                 V       : TMatrix); external 'dmath';
10006: { Unweighted fit of the acid-base titration curve }
10007: procedure WPKFit(X, Y, S : TVector;
10008:                  Lb, Ub  : Integer;
10009:                  MaxIter : Integer;
10010:                  Tol     : Float;
10011:                  B       : TVector;
10012:                  V       : TMatrix); external 'dmath';
10013: { Weighted fit of the acid-base titration curve }
10014: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10015: { Returns the value of the acid-base titration function at point X }
10016: procedure PowFit(X, Y    : TVector;
10017:                  Lb, Ub  : Integer;
10018:                  MaxIter : Integer;
10019:                  Tol     : Float;
10020:                  B       : TVector;
10021:                  V       : TMatrix); external 'dmath';
10022: { Unweighted fit of power function }
10023: procedure WPowFit(X, Y, S : TVector;
10024:                   Lb, Ub  : Integer;
10025:                   MaxIter : Integer;
10026:                   Tol     : Float;
10027:                   B       : TVector;
10028:                   V       : TMatrix); external 'dmath';
10029: { Weighted fit of power function }
10030:
10031: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10032: { Returns the value of the power function at point X }
10033: procedure GammaFit(X, Y    : TVector;
10034:                    Lb, Ub  : Integer;
10035:                    MaxIter : Integer;
10036:                    Tol     : Float;
10037:                    B       : TVector;
10038:                    V       : TMatrix); external 'dmath';
10039: { Unweighted fit of gamma distribution function }
10040: procedure WGammaFit(X, Y, S : TVector;
10041:                     Lb, Ub  : Integer;
10042:                     MaxIter : Integer;
10043:                     Tol     : Float;
10044:                     B       : TVector;
10045:                     V       : TMatrix); external 'dmath';
10046: { Weighted fit of gamma distribution function }
10047: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10048: { Returns the value of the gamma distribution function at point X }
10049: { ---------------------------------------------------------------
10050:   Principal component analysis
10051:   --------------------------------------------------------------- }
10052: procedure VecMean(X            : TMatrix;
10053:                   Lb, Ub, Nvar : Integer;
```

```
10054:                   M           : TVector); external 'dmath';
10055: { Computes the mean vector M from matrix X }
10056: procedure VecSD(X           : TMatrix;
10057:                 Lb, Ub, Nvar : Integer;
10058:                 M, S         : TVector); external 'dmath';
10059: { Computes the vector of standard deviations S from matrix X }
10060: procedure MatVarCov(X           : TMatrix;
10061:                     Lb, Ub, Nvar : Integer;
10062:                     M            : TVector;
10063:                     V            : TMatrix); external 'dmath';
10064: { Computes the variance-covariance matrix V from matrix X }
10065: procedure MatCorrel(V    : TMatrix;
10066:                     Nvar : Integer;
10067:                     R    : TMatrix); external 'dmath';
10068: { Computes the correlation matrix R from the var-cov matrix V }
10069: procedure PCA(R      : TMatrix;
10070:               Nvar   : Integer;
10071:               Lambda : TVector;
10072:               C, Rc  : TMatrix); external 'dmath';
10073: { Performs a principal component analysis of the correlation matrix R }
10074: procedure ScaleVar(X           : TMatrix;
10075:                    Lb, Ub, Nvar : Integer;
10076:                    M, S         : TVector;
10077:                    Z            : TMatrix); external 'dmath';
10078: { Scales a set of variables by subtracting means and dividing by SD's }
10079: procedure PrinFac(Z           : TMatrix;
10080:                   Lb, Ub, Nvar : Integer;
10081:                   C, F         : TMatrix); external 'dmath';
10082: { Computes principal factors }
10083: { ------------------------------------------------------------------
10084:   Strings
10085:   ------------------------------------------------------------------ }
10086: function LTrim(S : String) : String; external 'dmath';
10087: { Removes leading blanks }
10088: function RTrim(S : String) : String; external 'dmath';
10089: { Removes trailing blanks }
10090: function Trim(S : String) : String; external 'dmath';
10091: { Removes leading and trailing blanks }
10092: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10093: { Returns a string made of character C repeated N times }
10094: function RFill(S : String; L : Byte) : String; external 'dmath';
10095: { Completes string S with trailing blanks for a total length L }
10096: function LFill(S : String; L : Byte) : String; external 'dmath';
10097: { Completes string S with leading blanks for a total length L }
10098: function CFill(S : String; L : Byte) : String; external 'dmath';
10099: { Centers string S on a total length L }
10100: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10101: { Replaces in string S all the occurences of C1 by C2 }
10102: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10103: { Extracts a field from a string }
10104: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10105: { Parses a string into its constitutive fields }
10106: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10107: { Sets the numeric format }
10108: function FloatStr(X : Float) : String; external 'dmath';
10109: { Converts a real to a string according to the numeric format }
10110: function IntStr(N : LongInt) : String; external 'dmath';
10111: { Converts an integer to a string }
10112: function CompStr(Z : Complex) : String; external 'dmath';
10113: { Converts a complex number to a string }
10114: {$IFDEF DELPHI}
10115: function StrDec(S : String) : String; external 'dmath';
10116: { Set decimal separator to the symbol defined in SysUtils }
10117: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10118: { Test if a string represents a number and returns it in X }
10119: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10120: { Reads a floating point number from an Edit control }
10121: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10122: { Writes a floating point number in a text file }
10123: {$ENDIF}
10124: { ------------------------------------------------------------------
10125:   BGI / Delphi graphics
10126:   ------------------------------------------------------------------ }
10127: function InitGraphics
10128: {$IFDEF DELPHI}
10129: (Width, Height : Integer) : Boolean;
10130: {$ELSE}
10131: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10132: { Enters graphic mode }
10133: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10134:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10135: { Sets the graphic window }
10136: procedure SetOxScale(Scale               : TScale;
10137:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10138: { Sets the scale on the Ox axis }
10139: procedure SetOyScale(Scale               : TScale;
10140:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10141: { Sets the scale on the Oy axis }
10142: procedure GetOxScale(var Scale               : TScale;
```

```
10143:                     var OxMin, OxMax, OxStep : Float); external 'dmath';
10144: { Returns the scale on the Ox axis }
10145: procedure GetOyScale(var Scale               : TScale;
10146:                     var OyMin, OyMax, OyStep : Float); external 'dmath';
10147: { Returns the scale on the Oy axis }
10148: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10149: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10150: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10151: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10152: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10153: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10154: {$IFNDEF DELPHI}
10155: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10156: { Sets the font for the main graph title }
10157: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10158: { Sets the font for the Ox axis (title and labels) }
10159: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10160: { Sets the font for the Oy axis (title and labels) }
10161: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10162: { Sets the font for the legends }
10163: procedure SetClipping(Clip : Boolean); external 'dmath';
10164: { Determines whether drawings are clipped at the current viewport
10165:   boundaries, according to the value of the Boolean parameter Clip }
10166: {$ENDIF}
10167: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10168: { Plots the horizontal axis }
10169: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10170: { Plots the vertical axis }
10171: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10172: { Plots a grid on the graph }
10173: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10174: { Writes the title of the graph }
10175: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10176: { Sets the maximum number of curves and re-initializes their parameters }
10177: procedure SetPointParam
10178: {$IFDEF DELPHI}
10179: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10180: {$ELSE}
10181: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10182: { Sets the point parameters for curve # CurvIndex }
10183: procedure SetLineParam
10184: {$IFDEF DELPHI}
10185: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10186: {$ELSE}
10187: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10188: { Sets the line parameters for curve # CurvIndex }
10189: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10190: { Sets the legend for curve # CurvIndex }
10191: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10192: { Sets the step for curve # CurvIndex }
10193: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10194: procedure GetPointParam
10195: {$IFDEF DELPHI}
10196: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10197: {$ELSE}
10198: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10199: { Returns the point parameters for curve # CurvIndex }
10200: procedure GetLineParam
10201: {$IFDEF DELPHI}
10202: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10203: {$ELSE}
10204: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10205: { Returns the line parameters for curve # CurvIndex }
10206: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10207: { Returns the legend for curve # CurvIndex }
10208: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10209: { Returns the step for curve # CurvIndex }
10210: {$IFDEF DELPHI}
10211: procedure PlotPoint(Canvas    : TCanvas;
10212:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
10213: {$ELSE}
10214: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10215: {$ENDIF}
10216: { Plots a point on the screen }
10217: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10218:                     X, Y                 : TVector;
10219:                     Lb, Ub, CurvIndex    : Integer); external 'dmath';
10220: { Plots a curve }
10221: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10222:                                  X, Y, S              : TVector;
10223:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10224: { Plots a curve with error bars }
10225: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10226:                    Func                 : TFunc;
10227:                    Xmin, Xmax           : Float;
10228:                    {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10229:                    CurvIndex            : Integer); external 'dmath';
10230: { Plots a function }
10231: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
```

```
10232:                      NCurv                : Integer;
10233:                      ShowPoints, ShowLines : Boolean); external 'dmath';
10234: { Writes the legends for the plotted curves }
10235: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10236:                 Nx, Ny, Nc           : Integer;
10237:                 X, Y, Z              : TVector;
10238:                 F                    : TMatrix); external 'dmath';
10239: { Contour plot }
10240: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10241: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10242: function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
10243: function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
10244: {$IFNDEF DELPHI}
10245: procedure LeaveGraphics; external 'dmath';
10246: { Quits graphic mode }
10247: {$ENDIF}
10248: { ---------------------------------------------------------------
10249:   LaTeX graphics
10250:   --------------------------------------------------------------- }
10251: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
10252:                           Header            : Boolean) : Boolean; external 'dmath';
10253: { Initializes the LaTeX file }
10254: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10255: { Sets the graphic window }
10256: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10257: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10258: { Sets the scale on the Ox axis }
10259: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10260: { Sets the scale on the Oy axis }
10261: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10262: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10263: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10264: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10265: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10266: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10267: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10268: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10269: { Sets the maximum number of curves and re-initializes their parameters }
10270: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10271: { Sets the point parameters for curve # CurvIndex }
10272: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10273:                            Width : Float; Smooth : Boolean); external 'dmath';
10274: { Sets the line parameters for curve # CurvIndex }
10275: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10276: { Sets the legend for curve # CurvIndex }
10277: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10278: { Sets the step for curve # CurvIndex }
10279: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10280: { Plots a curve }
10281: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10282:                                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10283: { Plots a curve with error bars }
10284: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10285:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
10286: { Plots a function }
10287: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10288: { Writes the legends for the plotted curves }
10289: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
10290: { Contour plot }
10291: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10292: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10293:
10294: //*****************************************************unit uPSI_SynPdf;
10295:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10296:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10297:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10298:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10299:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10300:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10301:  //Function _GetCharCount( Text : PAnsiChar) : integer
10302:  //Procedure L2R( W : PWideChar; L : integer)
10303:  Function PdfCoord( MM : single) : integer
10304:  Function CurrentPrinterPaperSize : TPDFPaperSize
10305:  Function CurrentPrinterRes : TPoint
10306:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10307:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10308:  Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10309:  Const('Usp10','String 'usp10.dll
10310:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10311:    'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10312:   AddTypeS('TScriptState_set', 'set of TScriptState_enum
10313: //*****************************************************************
10314:
10315: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10316: begin
10317:  Procedure PMrandomize( I : word)
10318:  Function PMrandom : longint
10319:  Function Rrand : extended
10320:  Function Irand( N : word) : word
```

```
10321:  Function Brand( P : extended) : boolean
10322:  Function Nrand : extended
10323: end;
10324:
10325: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10326: begin
10327:   Function Endian( x : LongWord) : LongWord
10328:   Function Endian64( x : Int64) : Int64
10329:   Function spRol( x : LongWord; y : Byte) : LongWord
10330:   Function spRor( x : LongWord; y : Byte) : LongWord
10331:   Function Ror64( x : Int64; y : Byte) : Int64
10332: end;
10333:
10334: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10335: begin
10336:  Procedure ClearModules
10337:  Procedure ReadMapFile( Fname : string)
10338:  Function AddressInfo( Address : dword) : string
10339: end;
10340:
10341: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10342: begin
10343:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10344:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10345:    +'teByOther, tpExecuteByOther )
10346:   TTarPermissions', 'set of TTarPermission
10347:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10348:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10349:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10350:   TTarModes', 'set of TTarMode
10351:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10352:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10353:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10354:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10355:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10356:   SIRegister_TTarArchive(CL);
10357:   SIRegister_TTarWriter(CL);
10358:  Function PermissionString( Permissions : TTarPermissions) : STRING
10359:  Function ConvertFilename( Filename : STRING) : STRING
10360:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10361:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10362:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10363: end;
10364:
10365:
10366: //**************************************************unit uPSI_TlHelp32;
10367: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10368: begin
10369:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10370:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10371:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10372:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10373:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10374:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10375:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10376:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10377:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10378:   AddTypeS('THeapList32', 'tagHEAPLIST32
10379:  Const('HF32_DEFAULT','LongInt'( 1);
10380:  Const('HF32_SHARED','LongInt'( 2);
10381:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10382:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10383:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10384:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10385:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10386:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10387:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10388:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10389:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10390:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10391:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10392:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10393:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10394:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10395:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10396:    +'aPri : Longint; dwFlags : DWORD; end
10397:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10398:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10399:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10400:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10401: end;
10402:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10403:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10404:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10405:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10406:  Const('EWX_LOGOFF','LongInt'( 0);
10407:  Const('EWX_SHUTDOWN','LongInt'( 1);
10408:  Const('EWX_REBOOT','LongInt'( 2);
10409:  Const('EWX_FORCE','LongInt'( 4);
```

```
10410:  Const('EWX_POWEROFF','LongInt'( 8);
10411:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10412:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10413:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10414:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10415:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10416:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10417:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10418:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10419:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10420:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10421:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10422:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10423:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10424:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10425:  Function GetDesktopWindow : HWND
10426:  Function GetParent( hWnd : HWND) : HWND
10427:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10428:  Function GetTopWindow( hWnd : HWND) : HWND
10429:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10430:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10431:  //Delphi DFM
10432:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10433:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10434:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10435:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10436:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10437:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10438:  Function OpenIcon( hWnd : HWND) : BOOL
10439:  Function CloseWindow( hWnd : HWND) : BOOL
10440:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10441:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10442:  Function IsWindowVisible( hWnd : HWND) : BOOL
10443:  Function IsIconic( hWnd : HWND) : BOOL
10444:  Function AnyPopup : BOOL
10445:  Function BringWindowToTop( hWnd : HWND) : BOOL
10446:  Function IsZoomed( hWnd : HWND) : BOOL
10447:  Function IsWindow( hWnd : HWND) : BOOL
10448:  Function IsMenu( hMenu : HMENU) : BOOL
10449:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10450:  Function DestroyWindow( hWnd : HWND) : BOOL
10451:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10452:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10453:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10454:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10455:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10456:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10457:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10458:
10459: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10460: begin
10461:  const('ShowSetupDialogOptLong','String '--setup
10462:  PrimaryConfPathOptLong','String '--primary-config-path=
10463:  PrimaryConfPathOptShort','String '--pcp=
10464:  SecondaryConfPathOptLong','String '--secondary-config-path=
10465:  SecondaryConfPathOptShort','String '--scp=
10466:  NoSplashScreenOptLong','String '--no-splash-screen
10467:  NoSplashScreenOptShort','String '--nsc
10468:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10469:  SkipLastProjectOpt','String '--skip-last-project
10470:  DebugLogOpt','String '--debug-log=
10471:  DebugLogOptEnable','String '--debug-enable=
10472:  LanguageOpt','String '--language=
10473:  LazarusDirOpt','String '--lazarusdir=
10474:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10475:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10476:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10477:  Function IsHelpRequested : Boolean
10478:  Function IsVersionRequested : boolean
10479:  Function GetLanguageSpecified : string
10480:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10481:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10482:  Procedure ParseNoGuiCmdLineParams
10483:  Function ExtractCmdLineFilenames : TStrings
10484: end;
10485:
10486:
10487: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10488: begin
10489:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10490:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10491:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10492:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10493:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10494:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10495:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10496:  Function DirPathExists( DirectoryName : string) : boolean
10497:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10498:  Function ExtractFileNameOnly( const AFilename : string) : string
```

```
10499:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10500:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10501:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10502:  Function ForceDirectory( DirectoryName : string) : boolean
10503:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10504:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10505:  Function FileIsText( const AFilename : string) : boolean
10506:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10507:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10508:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10509:  Function TrimFilename( const AFilename : string) : string
10510:  Function ResolveDots( const AFilename : string) : string
10511:  Procedure ForcePathDelims( var FileName : string)
10512:  Function GetForcedPathDelims( const FileName : string) : String
10513:  Function CleanAndExpandFilename( const Filename : string) : string
10514:  Function CleanAndExpandDirectory( const Filename : string) : string
10515:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10516:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10517:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10518:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
10519:  Function FileIsInPath( const Filename, Path : string) : boolean
10520:  Function AppendPathDelim( const Path : string) : string
10521:  Function ChompPathDelim( const Path : string) : string
10522:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10523:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10524:  Function MinimizeSearchPath( const SearchPath : string) : string
10525:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10526:  (*Function FileExistsUTF8( const Filename : string) : boolean
10527:  Function FileAgeUTF8( const FileName : string) : Longint
10528:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10529:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10530:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10531:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10532:  Procedure FindCloseUTF8( var F : TSearchrec)
10533:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10534:  Function FileGetAttrUTF8( const FileName : String) : Longint
10535:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10536:  Function DeleteFileUTF8( const FileName : String) : Boolean
10537:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10538:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10539:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10540:  Function GetCurrentDirUTF8 : String
10541:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10542:  Function CreateDirUTF8( const NewDir : String) : Boolean
10543:  Function RemoveDirUTF8( const Dir : String) : Boolean
10544:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10545:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10546:  Function FileCreateUTF8( const FileName : string) : THandle;
10547:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10548:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10549:  Function FileSizeUtf8( const Filename : string) : int64
10550:  Function GetFileDescription( const AFilename : string) : string
10551:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10552:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10553:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10554:  Function IsUNCPath( const Path : String) : Boolean
10555:  Function ExtractUNCVolume( const Path : String) : String
10556:  Function ExtractFileRoot( FileName : String) : String
10557:  Function GetDarwinSystemFilename( Filename : string) : string
10558:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10559:  Function StrToCmdLineParam( const Param : string) : string
10560:  Function MergeCmdLineParams( ParamList : TStrings) : string
10561:  Procedure InvalidateFileStateCache( const Filename : string)
10562:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10563:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10564:  Function ReadFileToString( const Filename : string) : string
10565:  type
10566:   TCopyFileFlag = ( cffOverwriteFile,
10567:     cffCreateDestDirectory, cffPreserveTime );
10568:   TCopyFileFlags = set of TCopyFileFlag;*)
10569:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10570:   TCopyFileFlags', 'set of TCopyFileFlag
10571:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10572: end;
10573:
10574: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10575: begin
10576:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10577:   SIRegister_TMask(CL);
10578:   SIRegister_TParseStringList(CL);
10579:   SIRegister_TMaskList(CL);
10580:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10581:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10582:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10583:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10584: end;
10585:
```

```
10586: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10587: begin
10588:   //PShellHookInfo', '^TShellHookInfo // will not work
10589:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10590:   SHELLHOOKINFO', 'TShellHookInfo
10591:   LPSHELLHOOKINFO', 'PShellHookInfo
10592:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10593:   SIRegister_TJvShellHook(CL);
10594:  Function InitJvShellHooks : Boolean
10595:  Procedure UnInitJvShellHooks
10596: end;
10597:
10598: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10599: begin
10600:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10601:    +', dcHasSetSel, dcWantTab, dcNative )
10602:   TDlgCodes', 'set of TDlgCode
10603:  'dcWantMessage',' dcWantAllKeys);
10604:   SIRegister_IJvExControl(CL);
10605:   SIRegister_IJvDenySubClassing(CL);
10606:   SIRegister_TStructPtrMessage(CL);
10607:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10608:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10609:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10610:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10611:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10612:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10613:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10614:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10615:  Function GetFocusedControl( AControl : TControl) : TWinControl
10616:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10617:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10618:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10619:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10620:   SIRegister_TJvExControl(CL);
10621:   SIRegister_TJvExWinControl(CL);
10622:   SIRegister_TJvExCustomControl(CL);
10623:   SIRegister_TJvExGraphicControl(CL);
10624:   SIRegister_TJvExHintWindow(CL);
10625:   SIRegister_TJvExPubGraphicControl(CL);
10626: end;
10627:
10628: (*----------------------------------------------------------------------------*)
10629: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10630: begin
10631:  Procedure EncodeStream( Input, Output : TStream)
10632:  Procedure DecodeStream( Input, Output : TStream)
10633:  Function EncodeString1( const Input : string) : string
10634:  Function DecodeString1( const Input : string) : string
10635: end;
10636:
10637: (*----------------------------------------------------------------------------*)
10638: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10639: begin
10640:   SIRegister_TWebAppRegInfo(CL);
10641:   SIRegister_TWebAppRegList(CL);
10642:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10643:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10644:  Procedure UnregisterWebApp( const AProgID : string)
10645:  Function FindRegisteredWebApp( const AProgID : string) : string
10646:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10647:  'sUDPPort','String 'UDPPort
10648: end;
10649:
10650: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10651: begin
10652:  // TStringDynArray', 'array of string
10653:  Function GetEnvVarValue( const VarName : string) : string
10654:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10655:  Function DeleteEnvVar( const VarName : string) : Integer
10656:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10657:  Function ExpandEnvVars( const Str : string) : string
10658:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10659:  Procedure GetAllEnvVarNames( const Names : TStrings);
10660:  Function GetAllEnvVarNames1 : TStringDynArray;
10661:  Function EnvBlockSize : Integer
10662:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10663:   SIRegister_TPJEnvVarsEnumerator(CL);
10664:   SIRegister_TPJEnvVars(CL);
10665:   FindClass('TOBJECT'),'EPJEnvVars
10666:   FindClass('TOBJECT'),'EPJEnvVars
10667:  //Procedure Register
10668: end;
10669:
10670: (*----------------------------------------------------------------------------*)
10671: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10672: begin
10673:  'cOneSecInMS','LongInt'( 1000);
```

```
10674:  //'cDefTimeSlice','LongInt'( 50);
10675:  //'cDefMaxExecTime',' cOneMinInMS);
10676:  'cAppErrorMask','LongInt'( 1 shl 29);
10677:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10678:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10679:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10680:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10681:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10682:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10683:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10684:   SIRegister_TPJCustomConsoleApp(CL);
10685:   SIRegister_TPJConsoleApp(CL);
10686: end;
10687:
10688: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10689: begin
10690:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10691:   t_encoding', '( uuencode, base64, mime )
10692:  Function internet_date( date : TDateTime) : string
10693:  Function lookup_hostname( const hostname : string) : longint
10694:  Function my_hostname : string
10695:  Function my_ip_address : longint
10696:  Function ip2string( ip_address : longint) : string
10697:  Function resolve_hostname( ip : longint) : string
10698:  Function address_from( const s : string; count : integer) : string
10699:  Function encode_base64( data : TStream) : TStringList
10700:  Function decode_base64( source : TStringList) : TMemoryStream
10701:  Function posn( const s, t : string; count : integer) : integer
10702:  Function poscn( c : char; const s : string; n : integer) : integer
10703:  Function filename_of( const s : string) : string
10704:  //Function trim( const s : string) : string
10705:  //Procedure setlength( var s : string; l : byte)
10706:  Function TimeZoneBias : longint
10707:  Function eight2seven_quoteprint( const s : string) : string
10708:  Function eight2seven_german( const s : string) : string
10709:  Function seven2eight_quoteprint( const s : string) : string end;
10710:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10711:  Function socketerror : cint
10712:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10713:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10714:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10715:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10716:  Function fplisten( s : cint; backlog : cint) : cint
10717:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10718:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10719:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10720:  Function NetAddrToStr( Entry : in_addr) : String
10721:  Function HostAddrToStr( Entry : in_addr) : String
10722:  Function StrToHostAddr( IP : String) : in_addr
10723:  Function StrToNetAddr( IP : String) : in_addr
10724:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10725:   cint8', 'shortint
10726:   cuint8', 'byte
10727:   cchar', 'cint8
10728:   cschar', 'cint8
10729:   cuchar', 'cuint8
10730:   cint16', 'smallint
10731:   cuint16', 'word
10732:   cshort', 'cint16
10733:   csshort', 'cint16
10734:   cushort', 'cuint16
10735:   cint32', 'longint
10736:   cuint32', 'longword
10737:   cint', 'cint32
10738:   csint', 'cint32
10739:   cuint', 'cuint32
10740:   csigned', 'cint
10741:   cunsigned', 'cuint
10742:   cint64', 'int64
10743:   clonglong', 'cint64
10744:   cslonglong', 'cint64
10745:   cbool', 'longbool
10746:   cfloat', 'single
10747:   cdouble', 'double
10748:   clongdouble', 'extended
10749:
10750: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10751: begin
10752:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10753:   SIRegister_TlkJSONdotnetclass(CL);
10754:   SIRegister_TlkJSONbase(CL);
10755:   SIRegister_TlkJSONnumber(CL);
10756:   SIRegister_TlkJSONstring(CL);
10757:   SIRegister_TlkJSONboolean(CL);
10758:   SIRegister_TlkJSONnull(CL);
10759:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10760:    +'se; data : TObject; var Continue : Boolean)
10761:   SIRegister_TlkJSONcustomlist(CL);
10762:   SIRegister_TlkJSONlist(CL);
```

```
10763:   SIRegister_TlkJSONobjectmethod(CL);
10764:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10765:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10766:   SIRegister_TlkHashTable(CL);
10767:   SIRegister_TlkBalTree(CL);
10768:   SIRegister_TlkJSONobject(CL);
10769:   SIRegister_TlkJSON(CL);
10770:   SIRegister_TlkJSONstreamed(CL);
10771:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10772: end;
10773:
10774: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10775: begin
10776:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10777:   SIRegister_TZSortedList(CL);
10778:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10779:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10780:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10781:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10782:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10783:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10784:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10785:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10786:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10787:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10788:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10789:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10790:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10791:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10792:  Function StrToBoolEx( Str : string) : Boolean
10793:  Function BoolToStrEx( Bool : Boolean) : String
10794:  Function IsIpAddr( const Str : string) : Boolean
10795:  Function zSplitString( const Str, Delimiters : string) : TStrings
10796:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10797:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10798:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10799:  Function FloatToSQLStr( Value : Extended) : string
10800:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10801:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10802:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10803:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10804:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10805:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10806:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10807:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10808:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10809:  Function BytesToVar( const Value : TByteDynArray) : Variant
10810:  Function VarToBytes( const Value : Variant) : TByteDynArray
10811:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10812:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10813:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10814:  Function EncodeCString( const Value : string) : string
10815:  Function DecodeCString( const Value : string) : string
10816:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10817:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10818:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
10819:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
       SubVersion:Integer):Int;
10820:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10821:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10822:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10823:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10824:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10825:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10826:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10827:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10828:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10829:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10830: end;
10831:
10832: unit uPSI_ZEncoding;
10833:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10834:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10835:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10836:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10837:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10838:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10839:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10840:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10841:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10842:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10843:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10844:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10845:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10846:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10847:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10848:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10849:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
```

```
10850:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10851:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10852:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10853:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10854:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10855:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10856:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10857:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10858:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10859:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10860:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10861:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10862:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10863:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10864:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10865:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10866:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10867:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10868:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10869:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10870:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10871:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10872:  Function ZDefaultSystemCodePage : Word
10873:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10874:
10875:
10876: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10877: begin
10878:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10879:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10880:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10881:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10882:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10883:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10884:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10885:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10886:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10887:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10888:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
10889:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10890:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10891:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10892:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10893:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10894:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10895:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10896:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10897:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10898:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10899:  ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
10900:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10901:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10902:  ('EOAC_NONE','LongWord').SetUInt( $0);
10903:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10904:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10905:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10906:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10907:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10908:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10909:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10910:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10911:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10912:   FindClass('TOBJECT'),'EBoldCom
10913:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10914:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10915:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10916:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10917:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10918:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10919:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10920:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10921:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10922:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10923:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10924:  Function BoldCreateGUID : TGUID
10925:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
10926:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
10927:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10928:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10929: end;
10930:
10931: (*----------------------------------------------------------------------------*)
10932: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10933: begin
10934:  Function ParseISODate( s : string) : TDateTime
10935:  Function ParseISODateTime( s : string) : TDateTime
10936:  Function ParseISOTime( str : string) : TDateTime
10937: end;
```

```
10938:
10939: (*----------------------------------------------------------------------------*)
10940: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10941: begin
10942:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10943:  Function BoldCreateGUIDWithBracketsAsString : string
10944: end;
10945:
10946: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10947: begin
10948:   FindClass('TOBJECT'),'TBoldFileHandler
10949:   FindClass('TOBJECT'),'TBoldDiskFileHandler
10950:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
10951:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
10952:   SIRegister_TBoldFileHandler(CL);
10953:   SIRegister_TBoldDiskFileHandler(CL);
10954:  Procedure BoldCloseAllFilehandlers
10955:  Procedure BoldRemoveUnchangedFilesFromEditor
10956:  Function BoldFileHandlerList : TBoldObjectArray
10957:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
10958: end;
10959:
10960: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10961: begin
10962:   PCharArr', 'array of PChar
10963:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
10964:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
10965:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
10966:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
10967:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
10968:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
10969:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
10970:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
10971:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
10972:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
10973:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
       Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
10974:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
10975: end;
10976:
10977: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
10978: begin
10979:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
10980:   SIRegister_TfrmBoldQueryUser(CL);
10981:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
10982: end;
10983:
10984: (*----------------------------------------------------------------------------*)
10985: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
10986: begin
10987:  //('befIsInDisplayList',' BoldElementFlag0);
10988:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
10989:  //('befFollowerSelected',' BoldElementFlag2);
10990:   FindClass('TOBJECT'),'TBoldQueue
10991:   FindClass('TOBJECT'),'TBoldQueueable
10992:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
10993:   SIRegister_TBoldQueueable(CL);
10994:   SIRegister_TBoldQueue(CL);
10995:  Function BoldQueueFinalized : Boolean
10996:  Function BoldInstalledQueue : TBoldQueue
10997: end;
10998:
10999: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
11000: begin
11001:  const mmPerInch','Extended').setExtended( 25.4);
11002:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11003:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11004:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11005:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11006:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11007:   TBarLineType', '( white, black, black_half )
11008:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11009:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11010:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11011:   TCheckSumMethod', '( csmNone, csmModulo10 )
11012:   SIRegister_TAsBarcode(CL);
11013:  Function CheckSumModulo10( const data : string) : string
11014:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11015:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11016:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11017:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11018: end;
11019:
```

```
11020: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11021: begin
11022:    THomogeneousByteVector', 'array[0..3] of Byte
11023:   THomogeneousWordVector', 'array[0..3] of Word
11024:   THomogeneousIntVector', 'array[0..3] of Integer
11025:   THomogeneousFltVector', 'array[0..3] of single
11026:   THomogeneousDblVector', 'array[0..3] of double
11027:   THomogeneousExtVector', 'array[0..3] of extended
11028:   TAffineByteVector', 'array[0..2] of Byte
11029:   TAffineWordVector', 'array[0..2] of Word
11030:   TAffineIntVector', 'array[0..2] of Integer
11031:   TAffineFltVector', 'array[0..2] of single
11032:   TAffineDblVector', 'array[0..2] of double
11033:   TAffineExtVector', 'array[0..2] of extended
11034:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11035:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11036:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11037:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11038:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11039:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11040:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11041:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11042:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11043:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11044:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11045:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11046:   TMatrix4b', 'THomogeneousByteMatrix
11047:   TMatrix4w', 'THomogeneousWordMatrix
11048:   TMatrix4i', 'THomogeneousIntMatrix
11049:   TMatrix4f', 'THomogeneousFltMatrix
11050:   TMatrix4d', 'THomogeneousDblMatrix
11051:   TMatrix4e', 'THomogeneousExtMatrix
11052:   TMatrix3b', 'TAffineByteMatrix
11053:   TMatrix3w', 'TAffineWordMatrix
11054:   TMatrix3i', 'TAffineIntMatrix
11055:   TMatrix3f', 'TAffineFltMatrix
11056:   TMatrix3d', 'TAffineDblMatrix
11057:   TMatrix3e', 'TAffineExtMatrix
11058:   //'PMatrix', '^TMatrix // will not work
11059:   TMatrixGL', 'THomogeneousFltMatrix
11060:   THomogeneousMatrix', 'THomogeneousFltMatrix
11061:   TAffineMatrix', 'TAffineFltMatrix
11062:   TQuaternion', 'record Vector : TVector4f; end
11063:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11064:   +'ger; Height : Integer; end
11065:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11066:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11067:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11068:  'EPSILON','Extended').setExtended( 1E-100);
11069:  'EPSILON2','Extended').setExtended( 1E-50);
11070:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11071:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11072:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11073:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11074:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11075:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11076:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11077:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11078:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11079:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11080:  Function VectorLength( V : array of Single) : Single
11081:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11082:  Procedure VectorNegate( V : array of Single)
11083:  Function VectorNorm( V : array of Single) : Single
11084:  Function VectorNormalize( V : array of Single) : Single
11085:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11086:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11087:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11088:  Procedure VectorScale( V : array of Single; Factor : Single)
11089:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11090:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11091:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11092:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11093:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11094:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11095:  Procedure MatrixAdjoint( var M : TMatrixGL)
11096:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11097:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11098:  Function MatrixDeterminant( M : TMatrixGL) : Single
11099:  Procedure MatrixInvert( var M : TMatrixGL)
11100:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11101:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11102:  Procedure MatrixTranspose( var M : TMatrixGL)
11103:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11104:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11105:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11106:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11107:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11108:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
```

```
11109:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11110:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11111:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11112:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11113:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11114:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11115:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11116:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11117:  Function MakeAffineVector( V : array of Single) : TAffineVector
11118:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11119:  Function MakeVector( V : array of Single) : TVectorGL
11120:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11121:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11122:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11123:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11124:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11125:  Function ArcCosGL( X : Extended) : Extended
11126:  Function ArcSinGL( X : Extended) : Extended
11127:  Function ArcTan2GL( Y, X : Extended) : Extended
11128:  Function CoTanGL( X : Extended) : Extended
11129:  Function DegToRadGL( Degrees : Extended) : Extended
11130:  Function RadToDegGL( Radians : Extended) : Extended
11131:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11132:  Function TanGL( X : Extended) : Extended
11133:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11134:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11135:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11136:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11137:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11138:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11139: end;
11140:
11141:
11142: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11143: begin
11144:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11145:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11146:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11147:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11148:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11149:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11150:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11151:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11152:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11153:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11154:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11155:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11156:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11157:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11158:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11159:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11160:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11161:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11162:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11163:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11164:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11165:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11166:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11167:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11168:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11169:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
       SaveTo:TStrings):Bool;
11170:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11171: end;
11172:
11173: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11174: begin
11175:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11176:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11177:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11178:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11179:   FindClass('TOBJECT'),'EInvalidParam
11180:  Function IsDCOMInstalled : Boolean
11181:  Function IsDCOMEnabled : Boolean
11182:  Function GetDCOMVersion : string
11183:  Function GetMDACVersion : string
11184:  Function GetMDACVersion2 : string
11185:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11186:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11187:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11188:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11189:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11190:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11191:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11192:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
```

```
11193:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11194:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11195:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11196:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11197:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11198:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11199: end;
11200:
11201:
11202: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11203: begin
11204:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11205:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11206:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11207:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11208:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11209:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11210:  DegPerCycle','Extended').setExtended( 360.0);
11211:  DegPerGrad','Extended').setExtended( 0.9);
11212:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11213:  GradPerCycle','Extended').setExtended( 400.0);
11214:  GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
11215:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11216:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11217:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11218:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11219:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11220:  CyclePerGrad','Extended').setExtended( 0.0025);
11221:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11222:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11223:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11224:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11225:  Function MakePercentage( const Step, Max : Longint) : Longint
11226:  Function CelsiusToKelvin( const T : double) : double
11227:  Function CelsiusToFahrenheit( const T : double) : double
11228:  Function KelvinToCelsius( const T : double) : double
11229:  Function KelvinToFahrenheit( const T : double) : double
11230:  Function FahrenheitToCelsius( const T : double) : double
11231:  Function FahrenheitToKelvin( const T : double) : double
11232:  Function CycleToDeg( const Cycles : double) : double
11233:  Function CycleToGrad( const Cycles : double) : double
11234:  Function CycleToRad( const Cycles : double) : double
11235:  Function DegToCycle( const Degrees : double) : double
11236:  Function DegToGrad( const Degrees : double) : double
11237:  Function DegToRad( const Degrees : double) : double
11238:  Function GradToCycle( const Grads : double) : double
11239:  Function GradToDeg( const Grads : double) : double
11240:  Function GradToRad( const Grads : double) : double
11241:  Function RadToCycle( const Radians : double) : double
11242:  Function RadToDeg( const Radians : double) : double
11243:  Function RadToGrad( const Radians : double) : double
11244:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11245:  Function DmsToRad( const D, M : Integer; const S : double) : double
11246:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11247:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11248:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11249:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11250:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11251:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11252:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11253:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11254:  Function CmToInch( const Cm : double) : double
11255:  Function InchToCm( const Inch : double) : double
11256:  Function FeetToMetre( const Feet : double) : double
11257:  Function MetreToFeet( const Metre : double) : double
11258:  Function YardToMetre( const Yard : double) : double
11259:  Function MetreToYard( const Metre : double) : double
11260:  Function NmToKm( const Nm : double) : double
11261:  Function KmToNm( const Km : double) : double
11262:  Function KmToSm( const Km : double) : double
11263:  Function SmToKm( const Sm : double) : double
11264:  Function LitreToGalUs( const Litre : double) : double
11265:  Function GalUsToLitre( const GalUs : double) : double
11266:  Function GalUsToGalCan( const GalUs : double) : double
11267:  Function GalCanToGalUs( const GalCan : double) : double
11268:  Function GalUsToGalUk( const GalUs : double) : double
11269:  Function GalUkToGalUs( const GalUk : double) : double
11270:  Function LitreToGalCan( const Litre : double) : double
11271:  Function GalCanToLitre( const GalCan : double) : double
11272:  Function LitreToGalUk( const Litre : double) : double
11273:  Function GalUkToLitre( const GalUk : double) : double
11274:  Function KgToLb( const Kg : double) : double
11275:  Function LbToKg( const Lb : double) : double
11276:  Function KgToOz( const Kg : double) : double
11277:  Function OzToKg( const Oz : double) : double
11278:  Function CwtUsToKg( const Cwt : double) : double
11279:  Function CwtUkToKg( const Cwt : double) : double
11280:  Function KaratToKg( const Karat : double) : double
11281:  Function KgToCwtUs( const Kg : double) : double
```

```
11282:  Function KgToCwtUk( const Kg : double) : double
11283:  Function KgToKarat( const Kg : double) : double
11284:  Function KgToSton( const Kg : double) : double
11285:  Function KgToLton( const Kg : double) : double
11286:  Function StonToKg( const STon : double) : double
11287:  Function LtonToKg( const Lton : double) : double
11288:  Function QrUsToKg( const Qr : double) : double
11289:  Function QrUkToKg( const Qr : double) : double
11290:  Function KgToQrUs( const Kg : double) : double
11291:  Function KgToQrUk( const Kg : double) : double
11292:  Function PascalToBar( const Pa : double) : double
11293:  Function PascalToAt( const Pa : double) : double
11294:  Function PascalToTorr( const Pa : double) : double
11295:  Function BarToPascal( const Bar : double) : double
11296:  Function AtToPascal( const At : double) : double
11297:  Function TorrToPascal( const Torr : double) : double
11298:  Function KnotToMs( const Knot : double) : double
11299:  Function HpElectricToWatt( const HpE : double) : double
11300:  Function HpMetricToWatt( const HpM : double) : double
11301:  Function MsToKnot( const ms : double) : double
11302:  Function WattToHpElectric( const W : double) : double
11303:  Function WattToHpMetric( const W : double) : double
11304:  function getBigPI: string;    //PI of 1000 numbers
11305:
11306: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11307: begin
11308:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11309:  Procedure CDCopyFile( const FileName, DestName : string)
11310:  Procedure CDMoveFile( const FileName, DestName : string)
11311:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11312:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11313:  Function CDGetTempDir : string
11314:  Function CDGetFileSize( FileName : string) : longint
11315:  Function GetFileTime( FileName : string) : longint
11316:  Function GetShortName( FileName : string) : string
11317:  Function GetFullName( FileName : string) : string
11318:  Function WinReboot : boolean
11319:  Function WinDir : String
11320:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11321:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11322:  Function devExecutor : TdevExecutor
11323: end;
11324:
11325: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11326: begin
11327:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11328:  Procedure Associate( Index : integer)
11329:  Procedure UnAssociate( Index : integer)
11330:  Function IsAssociated( Index : integer) : boolean
11331:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11332:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11333:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11334:  procedure RefreshIcons;
11335: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11336: function MergColor(Colors: Array of TColor): TColor;
11337: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11338: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11339: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11340: function GetInverseColor(AColor: TColor): TColor;
11341: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11342: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11343: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11344: Procedure GetSystemMenuFont(Font: TFont);
11345: end;
11346:
11347: //**************************unit uPSI_JvHLParser;*****************************
11348: function IsStringConstant(const St: string): Boolean;
11349: function IsIntConstant(const St: string): Boolean;
11350: function IsRealConstant(const St: string): Boolean;
11351: function IsIdentifier(const ID: string): Boolean;
11352: function GetStringValue(const St: string): string;
11353: procedure ParseString(const S: string; Ss: TStrings);
11354: function IsStringConstantW(const St: WideString): Boolean;
11355: function IsIntConstantW(const St: WideString): Boolean;
11356: function IsRealConstantW(const St: WideString): Boolean;
11357: function IsIdentifierW(const ID: WideString): Boolean;
11358: function GetStringValueW(const St: WideString): WideString;
11359: procedure ParseStringW(const S: WideString; Ss: TStrings);
11360:
11361:
11362: //**************************unit uPSI_JclMapi;*****************************
11363:
11364: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11365: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11366: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11367: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
```

```
11368: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11369:
11370: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11371: begin
11372:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11373:  Function BuildType1Message( ADomain, AHost : String) : String
11374:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11375:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11376:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11377: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11378: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11379: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11380: end;
11381:
11382: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11383: begin
11384: ('IpAny','LongWord').SetUInt( $00000000);
11385:  IpLoopBack','LongWord').SetUInt( $7F000001);
11386:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11387:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11388:  PortAny','LongWord( $0000);
11389:  SocketMaxConnections','LongInt'( 5);
11390:  TIpAddr', 'LongWord
11391:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11392:  Function HostToNetLong( HostLong : LongWord) : LongWord
11393:  Function HostToNetShort( HostShort : Word) : Word
11394:  Function NetToHostLong( NetLong : LongWord) : LongWord
11395:  Function NetToHostShort( NetShort : Word) : Word
11396:  Function StrToIp( Ip : string) : TIpAddr
11397:  Function IpToStr( Ip : TIpAddr) : string
11398: end;
11399:
11400: (*---------------------------------------------------------------------------*)
11401: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11402: begin
11403:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11404:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11405:    +'amSha1, AlsmtpClientAuthAutoSelect )
11406:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11407:   SIRegister_TAlSmtpClient(CL);
11408: end;
11409:
11410: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11411: begin
11412: 'TBitNo', 'Integer
11413:   TStByteNo', 'Integer
11414: TStationNo', 'Integer
11415: TInOutNo', 'Integer
11416: TIo', '( EE, AA, NE, NA )
11417: TBitSet', 'set of TBitNo
11418: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11419: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11420: TBitAddr', 'LongInt
11421: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11422: TByteAddr', 'SmallInt
11423: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11424:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11425:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11426:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11427:  Function BitAddrToStr( Value : TBitAddr) : string
11428:  Function StrToBitAddr( const Value : string) : TBitAddr
11429:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11430:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11431:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11432:  Function ByteAddrToStr( Value : TByteAddr) : string
11433:  Function StrToByteAddr( const Value : string) : TByteAddr
11434:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11435:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11436:  Function InOutStateToStr( State : TInOutState) : string
11437:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11438:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11439: end;
11440:
11441: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11442: begin
11443: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11444:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11445: DpmiPmVector', 'Int64
11446:  'DInterval','LongInt'( 1000);
11447:  //'DEnabled','Boolean')BoolToStr( True);
11448:  'DIntFreq','string' if64
11449:  //'DMessages','Boolean if64);
11450:   SIRegister_TwdxCustomTimer(CL);
11451:   SIRegister_TwdxTimer(CL);
11452:   SIRegister_TwdxRtcTimer(CL);
11453:   SIRegister_TCustomIntTimer(CL);
11454:   SIRegister_TIntTimer(CL);
11455:   SIRegister_TRtcIntTimer(CL);
```

```
11456:  Function RealNow : TDateTime
11457:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11458:  Function DateTimeToMs( Time : TDateTime) : LongInt
11459: end;
11460:
11461: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11462: begin
11463: TIdSyslogPRI', 'Integer
11464: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11465:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11466:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11467:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11468:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11469: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11470:   SIRegister_TIdSysLogMsgPart(CL);
11471:   SIRegister_TIdSysLogMessage(CL);
11472:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11473:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11474:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11475:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11476:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11477:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11478: end;
11479:
11480: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11481: begin
11482:  'UWhitespace','String '(?:\s*)
11483:  Function StripSpaces( const AText : string) : string
11484:  Function CharCount( const AText : string; Ch : Char) : Integer
11485:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11486:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11487: end;
11488:
11489:
11490: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11491: begin
11492:  ExtPascalVersion','String '0.9.8
11493:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11494:    +'Opera, brKonqueror, brMobileSafari )
11495:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11496:   AddTypeS('TExtProcedure', 'Procedure
11497:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11498:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11499:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11500:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11501:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11502:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11503:  Function StrToJS( const S : string; UseBR : boolean) : string
11504:  Function CaseOf( const S : string; const Cases : array of string) : integer
11505:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11506:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11507:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11508:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11509:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11510:  Function IsUpperCase( S : string) : boolean
11511:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11512:  Function BeautifyCSS( const AStyle : string) : string
11513:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11514:  Function JSDateToDateTime( JSDate : string) : TDateTime
11515: end;
11516:
11517: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11518: begin
11519:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11520:   TSHDeleteOptions', 'set of TSHDeleteOption
11521:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11522:   TSHRenameOptions', 'set of TSHRenameOption
11523:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11524:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11525:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11526:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11527:   TEnumFolderFlags', 'set of TEnumFolderFlag
11528:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11529:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11530:    +'IEnumIdList; Folder : IShellFolder; end
11531:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11532:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11533:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11534:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11535:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11536:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11537:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11538:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11539:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11540:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11541:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11542:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11543:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11544:  Function SHFreeMem( var P : Pointer) : Boolean
```

```
11545:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11546:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11547:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11548:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11549:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11550:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11551:  Function PidlFree( var IdList : PItemIdList) : Boolean
11552:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11553:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11554:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11555:  Function PidlToPath( IdList : PItemIdList) : string
11556:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11557:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11558:   PShellLink', '^TShellLink // will not work
11559:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11560:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11561:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11562:  Procedure ShellLinkFree( var Link : TShellLink)
11563:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11564:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11565:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11566:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11567:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11568:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11569:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11570:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11571:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11572:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11573:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11574:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11575:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11576:  Function ShellOpenAs( const FileName : string) : Boolean
11577:  Function ShellRasDial( const EntryName : string) : Boolean
11578:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11579:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11580:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11581:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11582:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11583:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11584:  Function OemKeyScan( wOemChar : Word) : DWORD
11585:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11586: end;
11587:
11588: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11589: begin
11590:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11591:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11592:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11593:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11594:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11595:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11596:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11597:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11598:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11599:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11600:  Function xmlValidName( const Text : UnicodeString) : Boolean
11601:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11602:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11603:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11604:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11605:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11606:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11607:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11608:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11609:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11610:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11611:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11612:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11613:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11614:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11615:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11616:  Procedure SelfTestcXMLFunctions
11617: end;
11618:
11619: (*----------------------------------------------------------------------------*)
11620: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11621: begin
11622:  Function AWaitCursor : IUnknown
11623:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11624:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11625:  Function YesNo( const ACaption, AMsg : string) : boolean
11626:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11627:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11628:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11629:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11630:  Procedure GetSystemPaths( Strings : TStrings)
11631:  Procedure MakeEditNumeric( EditHandle : integer)
11632: end;
```

```
11633:
11634: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11635: begin
11636:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11637:  'BI_YUY2','LongWord( $32595559);
11638:  'BI_UYVY','LongWord').SetUInt( $59565955);
11639:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11640:  'BI_YVU9','LongWord').SetUInt( $39555659);
11641:  'BI_YUV12','LongWord( $30323449);
11642:  'BI_Y8','LongWord').SetUInt( $20203859);
11643:  'BI_Y211','LongWord').SetUInt( $31313259);
11644:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11645:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11646: end;
11647:
11648: (*----------------------------------------------------------------------------*)
11649: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11650: begin
11651:  'WM_USER','LongWord').SetUInt( $0400);
11652:  'WM_CAP_START','LongWord').SetUint($0400);
11653:  'WM_CAP_END','longword').SetUint($0400+85);
11654:  //WM_CAP_START+  85
11655:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11656:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11657:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11658:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11659:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11660:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11661:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11662:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11663:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11664:  Function capGetUserData( hwnd : THandle) : LongInt
11665:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11666:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11667:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11668:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11669:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11670:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11671:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11672:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11673:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11674:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11675:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11676:  Function capEditCopy( hwnd : THandle) : LongInt
11677:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11678:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11679:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11680:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11681:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11682:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11683:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11684:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11685:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11686:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11687:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11688:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11689:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11690:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11691:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11692:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11693:  Function capGrabFrame( hwnd : THandle) : LongInt
11694:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11695:  Function capCaptureSequence( hwnd : THandle) : LongInt
11696:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11697:  Function capCaptureStop( hwnd : THandle) : LongInt
11698:  Function capCaptureAbort( hwnd : THandle) : LongInt
11699:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11700:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11701:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11702:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11703:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11704:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11705:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11706:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11707:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11708:  Function capPalettePaste( hwnd : THandle) : LongInt
11709:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11710:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11711:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11712:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11713:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11714:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11715:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11716:   //PCapStatus', '^TCapStatus // will not work
11717:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11718:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11719:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11720:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11721:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
```

```
11722:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11723:    +' wNumAudioAllocated : WORD; end
11724:   //PCaptureParms', '^TCaptureParms // will not work
11725:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11726:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11727:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11728:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11729:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11730:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11731:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11732:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11733:    +'he : BOOL; AVStreamMaster : WORD; end
11734:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11735:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11736:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11737:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11738:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
       : Integer; hwndParent : THandle; nID : Integer) : THandle
11739:  Function
       capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11740:  'IDS_CAP_BEGIN','LongInt'( 300);
11741:  'IDS_CAP_END','LongInt'( 301);
11742:  'IDS_CAP_INFO','LongInt'( 401);
11743:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11744:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11745:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11746:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11747:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11748:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11749:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11750:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11751:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11752:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11753:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11754:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11755:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11756:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11757:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11758:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11759:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11760:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11761:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11762:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11763:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11764:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11765:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11766:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11767:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11768:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11769:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11770:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11771:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11772:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11773:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11774:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11775:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11776:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11777:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11778:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11779:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11780:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11781:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11782:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11783:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11784:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11785:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11786:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11787:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11788:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11789:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11790:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11791:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11792:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11793:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11794:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11795:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11796:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11797:  'AVICAP32','String 'AVICAP32.dll
11798: end;
11799:
11800: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11801: begin
11802:  Function AlBoolToInt( Value : Boolean) : Integer
11803:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11804:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11805:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11806:  Function ALInc( var x : integer; Count : integer) : Integer
11807:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11808:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
```

```
11809:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11810:  Function ALIsInteger(const S: AnsiString): Boolean;
11811:  function ALIsDecimal(const S: AnsiString): boolean;
11812:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11813:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11814:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11815:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11816:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11817:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11818:  Function ALRandomStr(const aLength: Longint): AnsiString;
11819:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11820:  Function ALRandomStrU(const aLength: Longint): String;
11821: end;
11822:
11823: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11824: begin
11825:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
       aTrueStr: AnsiString; const aFalseStr : AnsiString)
11826: end;
11827:
11828: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11829: begin
11830:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11831:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11832:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11833:    +'; ullAvailExtendedVirtual : Int64; end
11834:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11835:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11836:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11837:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11838:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11839:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11840:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11841:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11842: end;
11843:
11844: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11845: begin
11846:   SIRegister_THandledObject(CL);
11847:   SIRegister_TEvent(CL);
11848:   SIRegister_TMutex(CL);
11849:   SIRegister_TSharedMem(CL);
11850:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11851:  'TRACE_BUFFER','String 'TRACE_BUFFER
11852:  'TRACE_MUTEX','String 'TRACE_MUTEX
11853:   //PTraceEntry', '^TTraceEntry // will not work
11854:   SIRegister_TIPCTracer(CL);
11855:  'MAX_CLIENTS','LongInt'( 6);
11856:  'IPCTIMEOUT','LongInt'( 2000);
11857:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11858:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11859:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11860:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11861:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11862:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11863:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11864:   FindClass('TOBJECT'),'EMonitorActive
11865:   FindClass('TOBJECT'),'TIPCThread
11866:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11867:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11868:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11869:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11870:   TClientFlags', 'set of TClientFlag
11871:   //PEventData', '^TEventData // will not work
11872:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11873:    +'lag; Flags : TClientFlags; end
11874:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11875:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11876:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11877:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11878:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11879:   SIRegister_TIPCEvent(CL);
11880:   //PClientDirRecords', '^TClientDirRecords // will not work
11881:   SIRegister_TClientDirectory(CL);
11882:   TIPCState', '( stInActive, stDisconnected, stConnected )
11883:   SIRegister_TIPCThread(CL);
11884:   SIRegister_TIPCMonitor(CL);
11885:   SIRegister_TIPCClient(CL);
11886:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11887: end;
11888:
11889: (*----------------------------------------------------------------------------*)
11890: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11891: begin
11892:   SIRegister_TAlGSMComm(CL);
11893:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11894:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
       AMessage:AnsiString);
11895:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
```

```
11896:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
       UseGreekAlphabet:Bool):Widestring;
11897:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11898:  end;
11899:
11900: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11901: begin
11902:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11903:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11904:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11905:   TInternetScheme', 'integer
11906:   TALIPv6Binary', 'array[1..16] of Char;
11907:  // TALIPv6Binary = array[1..16] of ansiChar;
11908:  //   TInternetScheme = Integer;
11909:   SIRegister_TALHTTPRequestHeader(CL);
11910:   SIRegister_TALHTTPCookie(CL);
11911:   SIRegister_TALHTTPCookieCollection(CL);
11912:   SIRegister_TALHTTPResponseHeader(CL);
11913:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11914:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11915: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
       Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11916: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
       Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11917: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11918:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11919:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11920:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11921:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11922:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11923:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
       ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11924:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
       Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11925:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11926:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11927:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11928:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11929:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11930:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11931:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11932:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11933:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11934:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11935:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
11936:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11937:  Function ALZeroIpV6 : TALIPv6Binary
11938:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11939:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
11940:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11941:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
11942: end;
11943:
11944: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11945: begin
11946:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
11947:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11948:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
11949:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
11950:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
11951:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
11952:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
11953:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
11954:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
11955:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11956:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
11957: end;
11958:
11959: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11960: begin
11961:   SIRegister_TALEMailHeader(CL);
11962:   SIRegister_TALNewsArticleHeader(CL);
11963:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
11964:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
11965:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
11966:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
11967:  Function AlGenerateInternetMessageID : AnsiString;
11968:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
11969:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
11970:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11971: end;
11972:
11973: (*----------------------------------------------------------------------------*)
11974: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
```

```
11975: begin
11976:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11977:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
11978:  Function ALgetLocalIPs : TALStrings
11979:  Function ALgetLocalHostName : AnsiString
11980: end;
11981:
11982: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
11983: begin
11984:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
11985:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11986:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
11987:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
11988:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11989:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
11990:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
11991:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
11992:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
11993: end;
11994:
11995: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
11996: begin
11997:   TStartupInfoA', 'TStartupInfo
11998:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
11999:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
12000:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12001:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12002:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12003:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12004:  SE_TCB_NAME','String 'SeTcbPrivilege
12005:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12006:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12007:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12008:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12009:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12010:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12011:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12012:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12013:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12014:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12015:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12016:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12017:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12018:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12019:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12020:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12021:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12022:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12023:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12024:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12025:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12026:  Function AlGetEnvironmentString : AnsiString
12027:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12028:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12029:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12030:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12031:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12032: end;
12033:
12034: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12035: begin
12036:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12037:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12038:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12039:  Function ALGetModuleName : ansistring
12040:  Function ALGetModuleFileNameWithoutExtension : ansistring
12041:  Function ALGetModulePath : ansiString
12042:  Function AlGetFileSize( const AFileName : ansistring) : int64
12043:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12044:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12045:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12046:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12047:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12048:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12049:  Function ALFileExists( const Path : ansiString) : boolean
```

```
12050:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12051:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12052:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12053:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12054:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12055: end;
12056:
12057: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12058: begin
12059:   NativeInt', 'Integer
12060:   NativeUInt', 'Cardinal
12061:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12062:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12063:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12064:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12065:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12066:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12067:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12068:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12069:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12070:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12071:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12072:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12073:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12074:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12075:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12076:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12077:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12078:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12079:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12080:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12081:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12082:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12083:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12084:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12085:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12086:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12087:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12088:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12089:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12090:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12091:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12092:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12093: end;
12094:
12095: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12096: begin
12097:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12098:   FindClass('TOBJECT'),'TALXMLNode
12099:   FindClass('TOBJECT'),'TALXMLNodeList
12100:   FindClass('TOBJECT'),'TALXMLDocument
12101:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12102:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12103:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12104:    +'nst Name : AnsiString; const Attributes : TALStrings)
12105:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12106:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12107:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12108:    +'ntDocType, ntDocFragment, ntNotation )
12109:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12110:   TALXMLDocOptions', 'set of TALXMLDocOption
12111:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12112:   TALXMLParseOptions', 'set of TALXMLParseOption
12113:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12114:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12115:   SIRegister_EALXMLDocError(CL);
12116:   SIRegister_TALXMLNodeList(CL);
12117:   SIRegister_TALXMLNode(CL);
12118:   SIRegister_TALXmlElementNode(CL);
12119:   SIRegister_TALXmlAttributeNode(CL);
12120:   SIRegister_TALXmlTextNode(CL);
12121:   SIRegister_TALXmlDocumentNode(CL);
12122:   SIRegister_TALXmlCommentNode(CL);
12123:   SIRegister_TALXmlProcessingInstrNode(CL);
12124:   SIRegister_TALXmlCDataNode(CL);
12125:   SIRegister_TALXmlEntityRefNode(CL);
12126:   SIRegister_TALXmlEntityNode(CL);
```

```
12127:   SIRegister_TALXmlDocTypeNode(CL);
12128:   SIRegister_TALXmlDocFragmentNode(CL);
12129:   SIRegister_TALXmlNotationNode(CL);
12130:   SIRegister_TALXMLDocument(CL);
12131:  cAlXMLUTF8EncodingStr','String 'UTF-8
12132:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12133:  CALNSDelim','String ':
12134:  CALXML','String 'xml
12135:  CALVersion','String 'version
12136:  CALEncoding','String 'encoding
12137:  CALStandalone','String 'standalone
12138:  CALDefaultNodeIndent','String '
12139:  CALXmlDocument','String 'DOCUMENT
12140:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12141:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
       EncodingStr:AnsiString);
12142:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
       ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12143:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
       ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12144:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
       Recurse: Boolean):TalxmlNode
12145:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
       AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12146:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
       AnsiString
12147: end;
12148:
12149: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12150: //based on TEEProc, TeCanvas, TEEngine, TChart
12151: begin
12152:  'TeePiStep','Double').setExtended( Pi / 180.0);
12153:  'TeeDefaultPerspective','LongInt'( 100);
12154:  'TeeMinAngle','LongInt'( 270);
12155:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12156:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12157:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12158:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12159:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12160:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12161:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12162:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12163:  'TA_LEFT','LongInt'( 0);
12164:  'TA_RIGHT','LongInt'( 2);
12165:  'TA_CENTER','LongInt'( 6);
12166:  'TA_TOP','LongInt'( 0);
12167:  'TA_BOTTOM','LongInt'( 8);
12168:  'teePATCOPY','LongInt'( 0);
12169:  'NumCirclePoints','LongInt'( 64);
12170:  'teeDEFAULT_CHARSET','LongInt'( 1);
12171:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12172:  'TA_LEFT','LongInt'( 0);
12173:  'bs_Solid','LongInt'( 0);
12174:  'teepf24Bit','LongInt'( 0);
12175:  'teepfDevice','LongInt'( 1);
12176:  'CM_MOUSELEAVE','LongInt'( 10000);
12177:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12178:  'DC_BRUSH','LongInt'( 18);
12179:  'DC_PEN','LongInt'( 19);
12180:   teeCOLORREF', 'LongWord
12181:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12182:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12183:   SIRegister_TFilterRegion(CL);
12184:   SIRegister_IFormCreator(CL);
12185:   SIRegister_TTeeFilter(CL);
12186:   //TFilterClass', 'class of TTeeFilter
12187:   SIRegister_TFilterItems(CL);
12188:   SIRegister_TConvolveFilter(CL);
12189:   SIRegister_TBlurFilter(CL);
12190:   SIRegister_TTeePicture(CL);
12191:   TPenEndStyle', '( esRound, esSquare, esFlat )
12192:   SIRegister_TChartPen(CL);
12193:   SIRegister_TChartHiddenPen(CL);
12194:   SIRegister_TDottedGrayPen(CL);
12195:   SIRegister_TDarkGrayPen(CL);
12196:   SIRegister_TWhitePen(CL);
12197:   SIRegister_TChartBrush(CL);
12198:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12199:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12200:   SIRegister_TView3DOptions(CL);
12201:   FindClass('TOBJECT'),'TTeeCanvas
12202:   TTeeTransparency', 'Integer
12203:   SIRegister_TTeeBlend(CL);
12204:   FindClass('TOBJECT'),'TCanvas3D
12205:   SIRegister_TTeeShadow(CL);
12206:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
       gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12207:   FindClass('TOBJECT'),'TSubGradient
12208:   SIRegister_TCustomTeeGradient(CL);
```

```
12209:   SIRegister_TSubGradient(CL);
12210:   SIRegister_TTeeGradient(CL);
12211:   SIRegister_TTeeFontGradient(CL);
12212:   SIRegister_TTeeFont(CL);
12213:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12214:   TCanvasTextAlign', 'Integer
12215:   TTeeCanvasHandle', 'HDC
12216:   SIRegister_TTeeCanvas(CL);
12217:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12218:   SIRegister_TFloatXYZ(CL);
12219:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12220:   TRGB', 'record blue: byte; green: byte; red: byte; end
12221:   {TRGB=packed record
12222:     Blue  : Byte;
12223:     Green : Byte;
12224:     Red   : Byte;
12225:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12226:
12227: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12228:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12229: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12230: TCanvas3DPlane', '( cpX, cpY, cpZ )
12231:   //IInterface', 'IUnknown
12232:   SIRegister_TCanvas3D(CL);
12233:   SIRegister_TTeeCanvas3D(CL);
12234:   TTrianglePoints', 'Array[0..2] of TPoint;
12235:   TFourPoints', 'Array[0..3] of TPoint;
12236:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12237:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12238:  Function Point3D( const x, y, z : Integer) : TPoint3D
12239:  Procedure SwapDouble( var a, b : Double)
12240:  Procedure SwapInteger( var a, b : Integer)
12241:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12242:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12243:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12244:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12245:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12246:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12247:  Procedure UnClipCanvas( ACanvas : TCanvas)
12248:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12249:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12250:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12251:  'TeeCharForHeight','String 'W
12252:  'DarkerColorQuantity','Byte').SetUInt( 128);
12253:  'DarkColorQuantity','Byte').SetUInt( 64);
12254:   TButtonGetColorProc', 'Function  : TColor
12255:   SIRegister_TTeeButton(CL);
12256:   SIRegister_TButtonColor(CL);
12257:   SIRegister_TComboFlat(CL);
12258:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12259:  Function TeePoint( const aX, aY : Integer) : TPoint
12260:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12261:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12262:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12263:  Function OrientRectangle( const R : TRect) : TRect
12264:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12265:  Function PolygonBounds( const P : array of TPoint) : TRect
12266:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12267:  Function RGBValue( const Color : TColor) : TRGB
12268:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12269:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12270:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12271:  Function TeeCull( const P : TFourPoints) : Boolean;
12272:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12273:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12274:  Procedure SmoothStretch( Src, Dst : TBitmap);
12275:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12276:  Function TeeDistance( const x, y : Double) : Double
12277:  Function TeeLoadLibrary( const FileName : String) : HInst
12278:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12279:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12280:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12281:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
      Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12282:   SIRegister_ICanvasHyperlinks(CL);
12283:   SIRegister_ICanvasToolTips(CL);
12284:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12285: end;
12286:
12287: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12288: begin
12289:   TOvcHdc', 'Integer
12290:   TOvcHWND', 'Cardinal
12291:   TOvcHdc', 'HDC
12292:   TOvcHWND', 'HWND
12293:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12294:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12295:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12296:  Function DefaultEpoch : Integer
```

```
12297:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12298:  Procedure FixRealPrim( P : PChar; DC : Char)
12299:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12300:  Function GetLeftButton : Byte
12301:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12302:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12303:  Function GetShiftFlags : Byte
12304:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12305:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12306:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12307:  Function ovIsForegroundTask : Boolean
12308:  Function ovTrimLeft( const S : string) : string
12309:  Function ovTrimRight( const S : string) : string
12310:  Function ovQuotedStr( const S : string) : string
12311:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12312:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12313:  Function PtrDiff( const P1, P2 : PChar) : Word
12314:  Procedure PtrInc( var P, Delta : Word)
12315:  Procedure PtrDec( var P, Delta : Word)
12316:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12317:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
       SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12318:  Function ovMinI( X, Y : Integer) : Integer
12319:  Function ovMaxI( X, Y : Integer) : Integer
12320:  Function ovMinL( X, Y : LongInt) : LongInt
12321:  Function ovMaxL( X, Y : LongInt) : LongInt
12322:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12323:  Function PartialCompare( const S1, S2 : string) : Boolean
12324:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12325:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12326:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12327:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
      TransparentColor : TColor)
12328:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
      TransparentColor : TColorRef)
12329:  Function ovWidthOf( const R : TRect) : Integer
12330:  Function ovHeightOf( const R : TRect) : Integer
12331:  Procedure ovDebugOutput( const S : string)
12332:  Function GetArrowWidth( Width, Height : Integer) : Integer
12333:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12334:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12335:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12336:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12337:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12338:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12339:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12340:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12341:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12342:  Function CreateMetaFile( p1 : PChar) : HDC
12343:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12344:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12345:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12346:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12347:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12348:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12349:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12350:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12351:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12352:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12353:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12354:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12355:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
      SrcHeight:Int;Rop:DWORD):BOOL
12356:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12357:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
      SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12358:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12359:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12360:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12361:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12362:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12363:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12364:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12365:  Function UpdateColors( DC : HDC) : BOOL
12366:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12367:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12368:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12369:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12370:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12371:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12372:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
      HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12373:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
      yMask:Int):BOOL;
12374:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12375:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12376:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12377:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12378:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
```

```
12379:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12380:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12381:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12382:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12383:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12384:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12385:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12386: end;
12387:
12388: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12389: begin
12390:   SIRegister_TOvcAbstractStore(CL);
12391:   //PExPropInfo', '^TExPropInfo // will not work
12392: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12393:   SIRegister_TOvcPropertyList(CL);
12394:   SIRegister_TOvcDataFiler(CL);
12395:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12396:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12397:  Function CreateStoredItem( const CompName, PropName : string) : string
12398:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12399:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12400: end;
12401:
12402: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12403: begin
12404:  'ovsetsize','LongInt'( 16);
12405:  'etSyntax','LongInt'( 0);
12406:  'etSymantic','LongInt'( 1);
12407:  'chCR','Char #13);
12408:  'chLF','Char #10);
12409:  'chLineSeparator',' chCR);
12410:   SIRegister_TCocoError(CL);
12411:   SIRegister_TCommentItem(CL);
12412:   SIRegister_TCommentList(CL);
12413:   SIRegister_TSymbolPosition(CL);
12414: TGenListType', '( glNever, glAlways, glOnError )
12415: TovBitSet', 'set of Integer
12416:   //PStartTable', '^TStartTable // will not work
12417: 'TovCharSet', 'set of AnsiChar
12418: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12419: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12420: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12421: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12422: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12423:    +'osition; const Data : string; ErrorType : integer)
12424: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12425: TGetCH', 'Function ( pos : longint) : char
12426: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12427:   SIRegister_TCocoRScanner(CL);
12428:   SIRegister_TCocoRGrammar(CL);
12429:  '_EF','Char #0);
12430:  '_TAB','Char').SetString( #09);
12431:  '_CR','Char').SetString( #13);
12432:  '_LF','Char').SetString( #10);
12433:  '_EL','').SetString( _CR);
12434:  '_EOF','Char').SetString( #26);
12435:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12436:  'minErrDist','LongInt'( 2);
12437:  Function ovPadL( S : string; ch : char; L : integer) : string
12438: end;
12439:
12440:   TFormatSettings = record
12441:     CurrencyFormat: Byte;
12442:     NegCurrFormat: Byte;
12443:     ThousandSeparator: Char;
12444:     DecimalSeparator: Char;
12445:     CurrencyDecimals: Byte;
12446:     DateSeparator: Char;
12447:     TimeSeparator: Char;
12448:     ListSeparator: Char;
12449:     CurrencyString: string;
12450:     ShortDateFormat: string;
12451:     LongDateFormat: string;
12452:     TimeAMString: string;
12453:     TimePMString: string;
12454:     ShortTimeFormat: string;
12455:     LongTimeFormat: string;
12456:     ShortMonthNames: array[1..12] of string;
12457:     LongMonthNames: array[1..12] of string;
12458:     ShortDayNames: array[1..7] of string;
12459:     LongDayNames: array[1..7] of string;
12460:     TwoDigitYearCenturyWindow: Word;
12461:   end;
12462:
12463: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12464: begin
12465:  Function ovFormatSettings : TFormatSettings
12466: end;
12467:
```

```
12468: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12469: begin
12470:   TOvcCharSet', 'set of Char
12471:   ovBTable', 'array[0..255] of Byte
12472:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
       HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12473:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12474:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12475:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12476:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12477:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12478:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12479:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12480:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12481:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12482:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12483:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12484:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12485:  Function LoCaseChar( C : Char) : Char
12486:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12487:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12488:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12489:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12490:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12491:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12492:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12493:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12494:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12495:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12496:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12497:  Procedure TrimAllSpacesPChar( P : PChar)
12498:  Function TrimEmbeddedZeros( const S : string) : string
12499:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12500:  Function TrimTrailPrimPChar( S : PChar) : PChar
12501:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12502:  Function TrimTrailingZeros( const S : string) : string
12503:  Procedure TrimTrailingZerosPChar( P : PChar)
12504:  Function UpCaseChar( C : Char) : Char
12505:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12506:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12507:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12508: end;
12509:
12510: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12511: begin
12512:   //PRaiseFrame', '^TRaiseFrame // will not work
12513:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12514:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12515:  Procedure SafeCloseHandle( var Handle : THandle)
12516:  Procedure ExchangeInteger( X1, X2 : Integer)
12517:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12518:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12519:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12520:
12521: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12522:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12523: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12524:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12525:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12526:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12527:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12528:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12529:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12530:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12531:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12532:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12533:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12534:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12535:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12536:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12537:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12538:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12539:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12540:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12541:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12542:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12543:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12544:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12545:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12546:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12547:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12548:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12549:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12550:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12551:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12552:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12553:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12554:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12555:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
```

```
12556:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12557:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12558:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12559:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12560:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12561:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12562:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12563:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12564:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12565:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12566:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12567:       var lpLuid: TLargeInteger): BOOL; stdcall;
12568:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12569:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12570:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12571:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12572:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12573:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12574:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12575:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12576:       var GenerateOnClose: BOOL): BOOL; stdcall;
12577:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12578:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12579:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12580:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12581:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12582:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12583:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12584:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12585:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12586:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12587:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12588:       var phkResult: HKEY): Longint; stdcall;
12589:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12590:       var phkResult: HKEY): Longint; stdcall;
12591:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12592:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12593:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12594:       lpdwDisposition: PDWORD): Longint; stdcall;
12595:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12596:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12597:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12598:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12599:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12600:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12601:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12602:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12603:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12604:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12605:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12606:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12607:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12608:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12609:       lpcbClass: PDWORD; lpReserved: Pointer;
12610:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12611:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12612:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12613:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12614:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12615:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12616:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12617:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12618:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12619:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12620:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12621:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12622:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12623:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12624:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12625:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12626:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12627:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12628:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12629:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12630:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12631:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12632:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12633:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12634:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12635:
12636:  Function wAddAtom( lpString : PKOLChar) : ATOM
12637:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12638:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12639:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12640:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
     lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12641:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12642:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
```

```
12643:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12644:  Function wCreateDirectoryEx(lpTemplateDirectory,
        lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12645:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
        bInitialState:BOOL;lpName:PKOLChar):THandle;
12646:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
        PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12647:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
        dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12648:  Function wCreateHardLink(lpFileName,
        lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12649:  Function
        CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12650:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
        nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12651:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
        lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
        Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
        lpProcessInfo:TProcessInformation):BOOL
12652:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
        Longint; lpName : PKOLChar) : THandle
12653:  Function
        wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12654:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12655:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12656:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12657:  //Function
        wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12658:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12659:  //Function
        wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12660:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12661:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12662:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12663:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12664:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12665:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12666:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
        dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12667:  Function wFindAtom( lpString : PKOLChar) : ATOM
12668:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12669:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12670:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12671:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12672:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12673:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12674:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12675:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12676:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12677:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12678:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12679:  Function wGetCommandLine : PKOLChar
12680:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12681:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12682:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12683:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12684:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12685:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12686:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12687:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12688:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12689:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12690:  Function wGetEnvironmentStrings : PKOLChar
12691:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12692:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12693:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12694:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12695:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12696:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12697:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12698:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12699:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12700:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12701:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12702:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12703:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
```

```
12704:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
       nSize:DWORD; lpFileName : PKOLChar) : DWORD
12705:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12706:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12707:  Function wGetProfileString(lpAppName,lpKeyName,
       lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12708:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12709:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12710: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
       lpCharType):BOOL
12711:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12712:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12713:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12714: //Function
       wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12715: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12716: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
       : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
       lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12717:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12718:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12719:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12720:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12721:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12722:  Function
       wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12723:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12724:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12725:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12726:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12727:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
       TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12728:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12729:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12730:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12731:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12732:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12733:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12734:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
       lpNumberOfEventsRead:DWORD):BOOL;
12735:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12736:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12737:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
       lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12738:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
       lpNumbOfEventsRead:DWORD):BOOL;
12739:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
       : TCoord; var lpReadRegion : TSmallRect) : BOOL
12740:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
       DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12741:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12742:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
       lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12743:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
       lpFilePart:PKOLChar):DWORD;
12744:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12745:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12746:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12747:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12748:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12749:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12750:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12751:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12752:  //Function wUpdateResource(hUpdate:THandle;lpType,
       lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12753:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12754:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12755:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
       DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12756:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
       var lpNumberOfEventsWritten : DWORD) : BOOL
12757:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
       TCoord; var lpWriteRegion : TSmallRect) : BOOL
12758:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
       DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12759:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12760:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12761:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12762:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12763:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12764:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12765:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12766:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12767:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12768:  Function wlstrlen( lpString : PKOLChar) : Integer
12769:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
       PNetConnectInfoStruct) : DWORD
12770:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
       lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
```

```
12771:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12772:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12773:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12774:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12775:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12776:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12777:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12778:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12779:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
        : PKOLChar; nNameBufSize : DWORD) : DWORD
12780:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12781:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12782:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12783: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
12784:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12785: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12786: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12787:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12788:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12789:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12790:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12791:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12792: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12793: //Function wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12794: //Function wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12795:  Function wAddFontResource( FileName : PKOLChar) : Integer
12796:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12797:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12798:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12799:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12800:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12801: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12802:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
        fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
        fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12803:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12804:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12805: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12806:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12807:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12808: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
        pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12809: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12810:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12811:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12812:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12813:  //Function wExtTextOut(DC:HDC;X,
        Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12814:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12815:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12816:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12817:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12818: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12819: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12820:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12821:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12822: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12823: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
        lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12824:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12825: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12826:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12827: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12828:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12829:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12830:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12831:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12832:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12833:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12834:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12835:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12836:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12837:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12838:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12839:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12840:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12841:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12842:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12843:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12844:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12845:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
```

```
12846:  //Function
        wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12847:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12848: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
        dwFlags : DWORD; lParam : Pointer) : Longint
12849:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12850:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12851:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12852:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12853:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12854:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12855: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12856:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12857:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12858:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12859:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12860:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12861:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12862:  //Function wCreateDesktop(lpszDesktop,
        lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12863:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
        HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12864:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
        lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12865:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
        hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12866: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
        nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12867:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
        dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12868:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12869:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12870:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12871:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12872:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
        : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12873:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
        : TFNDlgProc; dwInitParam : LPARAM) : Integer
12874:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12875:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
        nIDStaticPath:Integer;uFileType:UINT):Integer;
12876:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
        nIDStaticPath:Int;uFiletype:UINT):Int;
12877:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12878:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12879: //Function wDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
        cy:Int;Flags:UINT):BOOL;
12880:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12881:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12882:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12883:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
        pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12884: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12885:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12886:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12887:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12888:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12889:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12890:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12891:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12892:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12893:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12894:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12895:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12896:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
        lpnTabStopPositions) : DWORD
12897: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
        lpnLengthNeed:DWORD)BOOL;
12898:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12899:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12900:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12901:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12902: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
        nHeigt:Int):BOOL;
12903:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12904:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12905:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12906:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12907:  Function wIsCharLower( ch : KOLChar) : BOOL
12908:  Function wIsCharUpper( ch : KOLChar) : BOOL
12909:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12910:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12911:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12912:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12913:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12914:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12915:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12916:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12917:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
```

```
12918:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12919:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12920:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12921:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
12922:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
12923:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12924:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
12925:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12926:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
12927:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
12928:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12929:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12930:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12931:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
12932:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
12933:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12934:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12935:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
12936: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
12937: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
12938:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
12939: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12940:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
12941:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
12942:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12943:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12944:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
      lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
12945:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
      lpdwResult:DWORD): LRESULT
12946:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12947:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
12948:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
12949:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12950:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
12951: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
12952:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
12953:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
12954:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
12955:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12956: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
12957:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
      lpnTabStopPositions,nTabOrigin:Int):Longint;
12958:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
12959:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
12960:  Function wVkKeyScan( ch : KOLChar) : SHORT
12961:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
12962:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
12963:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
12964:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
12965:
12966: //TestDrive!
12967: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
12968:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
12969:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
12970:  Function GetLocalUserSidStr( const UserName : string) : string
12971:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
12972:  Function Impersonate2User( const domain : string; const user : string) : boolean
12973:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
12974:  Function KillProcessbyname( const exename : string; var found : integer) : integer
12975:  Function getWinProcessList : TStringList
12976: end;
12977:
12978: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
12979: begin
12980:  'AfMaxSyncSlots','LongInt'( 64);
12981:  'AfSynchronizeTimeout','LongInt'( 2000);
12982:   TAfSyncSlotID', 'DWORD
12983:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
12984:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
12985:   TAfSafeDirectSyncEvent', 'Procedure
12986:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
12987:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
12988:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
12989:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
12990:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
12991:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
12992:  Function AfIsSyncMethod : Boolean
12993:  Function AfSyncWnd : HWnd
12994:  Function AfSyncStatistics : TAfSyncStatistics
12995:  Procedure AfClearSyncStatistics
12996: end;
12997:
12998: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
12999: begin
13000:  'fBinary','LongWord')( $00000001);
13001:  'fParity','LongWord'( $00000002);
13002:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13003:  'fOutxDsrFlow','LongWord')( $00000008);
```

```
13004:  'fDtrControl','LongWord')( $00000030);
13005:  'fDtrControlDisable','LongWord')( $00000000);
13006:  'fDtrControlEnable','LongWord')( $00000010);
13007:  'fDtrControlHandshake','LongWord')( $00000020);
13008:  'fDsrSensitivity','LongWord')( $00000040);
13009:  'fTXContinueOnXoff','LongWord')( $00000080);
13010:  'fOutX','LongWord')( $00000100);
13011:  'fInX','LongWord')( $00000200);
13012:  'fErrorChar','LongWord')( $00000400);
13013:  'fNull','LongWord')( $00000800);
13014:  'fRtsControl','LongWord')( $00003000);
13015:  'fRtsControlDisable','LongWord')( $00000000);
13016:  'fRtsControlEnable','LongWord')( $00001000);
13017:  'fRtsControlHandshake','LongWord')( $00002000);
13018:  'fRtsControlToggle','LongWord')( $00003000);
13019:  'fAbortOnError','LongWord')( $00004000);
13020:  'fDummy2','LongWord')( $FFFF8000);
13021:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13022:   FindClass('TOBJECT'),'EAfComPortCoreError
13023:   FindClass('TOBJECT'),'TAfComPortCore
13024:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13025:    +'tKind : TAfCoreEvent; Data : DWORD)
13026:   SIRegister_TAfComPortCoreThread(CL);
13027:   SIRegister_TAfComPortEventThread(CL);
13028:   SIRegister_TAfComPortWriteThread(CL);
13029:   SIRegister_TAfComPortCore(CL);
13030:  Function FormatDeviceName( PortNumber : Integer) : string
13031: end;
13032:
13033: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13034: begin
13035:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13036:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13037:   SIRegister_TApplicationFileIO(CL);
13038:   TDataFileCapability', '( dfcRead, dfcWrite )
13039:   TDataFileCapabilities', 'set of TDataFileCapability
13040:   SIRegister_TDataFile(CL);
13041:   //TDataFileClass', 'class of TDataFile
13042:  Function ApplicationFileIODefined : Boolean
13043:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13044:  Function FileStreamExists(const fileName: String) : Boolean
13045:  //Procedure Register
13046: end;
13047:
13048: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13049: begin
13050:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13051:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13052:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13053:   TALFBXScale', 'Integer
13054:   FindClass('TOBJECT'),'EALFBXConvertError
13055:   SIRegister_EALFBXError(CL);
13056:   SIRegister_EALFBXException(CL);
13057:   FindClass('TOBJECT'),'EALFBXGFixError
13058:   FindClass('TOBJECT'),'EALFBXDSQLError
13059:   FindClass('TOBJECT'),'EALFBXDynError
13060:   FindClass('TOBJECT'),'EALFBXGBakError
13061:   FindClass('TOBJECT'),'EALFBXGSecError
13062:   FindClass('TOBJECT'),'EALFBXLicenseError
13063:   FindClass('TOBJECT'),'EALFBXGStatError
13064:   //EALFBXExceptionClass', 'class of EALFBXError
13065:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13066:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13067:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13068:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13069:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13070:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13071:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13072:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13073:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13074:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13075:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13076:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13077:   TALFBXTransParams', 'set of TALFBXTransParam
13078:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13079:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13080:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13081: 'cALFBXMaxParamLength','LongInt'( 125);
13082:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13083:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13084:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13085:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13086:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13087:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13088:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13089:   SIRegister_TALFBXSQLDA(CL);
13090:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13091:   SIRegister_TALFBXPoolStream(CL);
13092:   //PALFBXBlobData', '^TALFBXBlobData // will not work
```

```
13093:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13094:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13095:   //TALFBXArrayDesc', 'TISCArrayDesc
13096:   //TALFBXBlobDesc', 'TISCBlobDesc
13097:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13098:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13099:   SIRegister_TALFBXSQLResult(CL);
13100:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13101:   SIRegister_TALFBXSQLParams(CL);
13102:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13103:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13104:    +'atementType : TALFBXStatementType; end
13105:   FindClass('TOBJECT'),'TALFBXLibrary
13106:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13107:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13108:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13109:    //+' Excep : EALFBXExceptionClass)
13110:   SIRegister_TALFBXLibrary(CL);
13111:  'cAlFBXDateOffset','LongInt'( 15018);
13112:  'cALFBXTimeCoeff','LongInt'( 864000000);
13113:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13114:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13115:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13116:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13117:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13118:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13119:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13120:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13121:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13122:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13123:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13124:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13125:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13126:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13127: end;
13128:
13129: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13130: begin
13131:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13132:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13133:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13134:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13135:    +'teger; First : Integer; CacheThreshold : Integer; end
13136:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13137:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13138:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13139:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13140:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13141:   SIRegister_TALFBXClient(CL);
13142:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13143:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13144:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13145:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13146:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13147:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13148:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13149:   SIRegister_TALFBXConnectionPoolClient(CL);
13150:   SIRegister_TALFBXEventThread(CL);
13151:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13152: end;
13153:
13154: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13155: begin
13156: _OSVERSIONINFOA = record
13157:     dwOSVersionInfoSize: DWORD;
13158:     dwMajorVersion: DWORD;
13159:     dwMinorVersion: DWORD;
13160:     dwBuildNumber: DWORD;
13161:     dwPlatformId: DWORD;
13162:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13163:   end;
13164:  TOSVersionInfoA', '_OSVERSIONINFOA
13165:   TOSVersionInfo', 'TOSVersionInfoA
13166:  'WS_EX_RIGHT','LongWord')( $00001000);
13167:  'WS_EX_LEFT','LongWord')( $00000000);
13168:  'WS_EX_RTLREADING','LongWord')( $00002000);
13169:  'WS_EX_LTRREADING','LongWord')( $00000000);
13170:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13171:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13172:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13173:  'LAYOUT_RTL','LongWord')( $00000001);
13174:  'LAYOUT_BTT','LongWord')( $00000002);
13175:  'LAYOUT_VBH','LongWord')( $00000004);
13176:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13177:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13178:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13179:  Function GetLayout( dc : hdc) : DWORD
13180:  Function IsBidi : Boolean
13181:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
```

```
13182:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13183:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13184:  Function GetPriorityClass( hProcess : THandle) : DWORD
13185:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13186:  Function CloseClipboard : BOOL
13187:  Function GetClipboardSequenceNumber : DWORD
13188:  Function GetClipboardOwner : HWND
13189:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13190:  Function GetClipboardViewer : HWND
13191:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13192:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13193:  Function GetClipboardData( uFormat : UINT) : THandle
13194:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13195:  Function CountClipboardFormats : Integer
13196:  Function EnumClipboardFormats( format : UINT) : UINT
13197:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13198:  Function EmptyClipboard : BOOL
13199:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13200:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13201:  Function GetOpenClipboardWindow : HWND
13202:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13203:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13204:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13205:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13206:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13207:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13208:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13209:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13210:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13211: end;
13212:
13213: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13214: begin
13215:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13216:  Function GetTemporaryFilesPath : String
13217:  Function GetTemporaryFileName : String
13218:  Function FindFileInPaths( const fileName, paths : String) : String
13219:  Function PathsToString( const paths : TStrings) : String
13220:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13221:  //Function MacroExpandPath( const aPath : String) : String
13222: end;
13223:
13224: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13225: begin
13226:   SIRegister_TALMultiPartBaseContent(CL);
13227:   SIRegister_TALMultiPartBaseContents(CL);
13228:   SIRegister_TAlMultiPartBaseStream(CL);
13229:   SIRegister_TALMultipartBaseEncoder(CL);
13230:   SIRegister_TALMultipartBaseDecoder(CL);
13231:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13232:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13233:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13234: end;
13235:
13236: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13237: begin
13238:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13239:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13240:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13241:  Function aAllocPadedMem( Size : Cardinal) : TObject
13242:  Procedure aFreePadedMem( var P : TObject);
13243:  Procedure aFreePadedMem1( var P : PChar);
13244:  Function aCheckPadedMem( P : Pointer) : Byte
13245:  Function aGetPadMemSize( P : Pointer) : Cardinal
13246:  Function aAllocMem( Size : Cardinal) : Pointer
13247:  Function aStrLen( const Str : PChar) : Cardinal
13248:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13249:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13250:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13251:  Function aStrEnd( const Str : PChar) : PChar
13252:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13253:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13254:  Function aPCharLength( const Str : PChar) : Cardinal
13255:  Function aPCharUpper( Str : PChar) : PChar
13256:  Function aPCharLower( Str : PChar) : PChar
13257:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13258:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13259:  Function aCopyTail( const S : String; Len : Integer) : String
13260:  Function aInt2Thos( I : Int64) : String
13261:  Function aUpperCase( const S : String) : String
13262:  Function aLowerCase( const S : string) : String
13263:  Function aCompareText( const S1, S2 : string) : Integer
13264:  Function aSameText( const S1, S2 : string) : Boolean
13265:  Function aInt2Str( Value : Int64) : String
13266:  Function aStr2Int( const Value : String) : Int64
13267:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13268:  Function aGetFileExt( const FileName : String) : String
13269:  Function aGetFilePath( const FileName : String) : String
13270:  Function aGetFileName( const FileName : String) : String
```

```
13271:  Function aChangeExt( const FileName, Extension : String) : String
13272:  Function aAdjustLineBreaks( const S : string) : string
13273:  Function aGetWindowStr( WinHandle : HWND) : String
13274:  Function aDiskSpace( Drive : String) : TdriveSize
13275:  Function aFileExists( FileName : String) : Boolean
13276:  Function aFileSize( FileName : String) : Int64
13277:  Function aDirectoryExists( const Name : string) : Boolean
13278:  Function aSysErrorMessage( ErrorCode : Integer) : string
13279:  Function aShortPathName( const LongName : string) : string
13280:  Function aGetWindowVer : TWinVerRec
13281:  procedure InitDriveSpacePtr;
13282: end;
13283:
13284: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13285: begin
13286:  aZero','LongInt'( 0);
13287:  'makeappDEF','LongInt'( - 1);
13288:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13289:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13290:  'CS_KEYCVTWINDOW','LongInt'( 4);
13291:  'CS_DBLCLKS','LongInt'( 8);
13292:  'CS_OWNDC','LongWord')( $20);
13293:  'CS_CLASSDC','LongWord')( $40);
13294:  'CS_PARENTDC','LongWord')( $80);
13295:  'CS_NOKEYCVT','LongWord')( $100);
13296:  'CS_NOCLOSE','LongWord')( $200);
13297:  'CS_SAVEBITS','LongWord')( $800);
13298:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13299:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13300:  'CS_GLOBALCLASS','LongWord')( $4000);
13301:  'CS_IME','LongWord')( $10000);
13302:  'CS_DROPSHADOW','LongWord')( $20000);
13303:   //PPanelFunc', '^TPanelFunc // will not work
13304:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13305:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13306:   TFontLooks', 'set of TFontLook
13307:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13308:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13309:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13310:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13311:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13312:  Procedure RunMsgLoop( Show : Boolean)
13313:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13314:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
       ID_Number:Cardinal;hFont:Int):Int;
13315:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13316:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13317:  Function MakePanel(Left,Top,Width,Height,
       hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13318:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13319:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13320:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13321: end;
13322:
13323: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13324: begin
13325:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13326:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13327:   TScreenSaverOptions', 'set of TScreenSaverOption
13328:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13329:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13330:   SIRegister_TScreenSaver(CL);
13331:  //Procedure Register
13332:  Procedure SetScreenSaverPassword
13333: end;
13334:
13335: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13336: begin
13337:   FindClass('TOBJECT'),'TXCollection
13338:   SIRegister_EFilerException(CL);
13339:   SIRegister_TXCollectionItem(CL);
13340:   //TXCollectionItemClass', 'class of TXCollectionItem
13341:   SIRegister_TXCollection(CL);
13342:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13343:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13344:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13345:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13346:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13347:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13348: end;
13349:
13350: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13351: begin
13352:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13353:  Procedure xglMapTexCoordToNull
13354:  Procedure xglMapTexCoordToMain
13355:  Procedure xglMapTexCoordToSecond
13356:  Procedure xglMapTexCoordToDual
```

```
13357:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13358:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13359:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13360:  Procedure xglBeginUpdate
13361:  Procedure xglEndUpdate
13362:  Procedure xglPushState
13363:  Procedure xglPopState
13364:  Procedure xglForbidSecondTextureUnit
13365:  Procedure xglAllowSecondTextureUnit
13366:  Function xglGetBitWiseMapping : Cardinal
13367: end;
13368:
13369: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13370: begin
13371:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13372:   TBaseListOptions', 'set of TBaseListOption
13373:   SIRegister_TBaseList(CL);
13374:   SIRegister_TBaseVectorList(CL);
13375:   SIRegister_TAffineVectorList(CL);
13376:   SIRegister_TVectorList(CL);
13377:   SIRegister_TTexPointList(CL);
13378:   SIRegister_TXIntegerList(CL);
13379:   //PSingleArrayList', '^TSingleArrayList // will not work
13380:   SIRegister_TSingleList(CL);
13381:   SIRegister_TByteList(CL);
13382:   SIRegister_TQuaternionList(CL);
13383:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13384:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13385:  Procedure FastQuickSortLists(startIndex,
      endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13386: end;
13387:
13388: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13389: begin
13390:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13391:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13392:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
      indices:TIntegerList;list:TAffineVectorList);
13393:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
      indices:TIntegerList;list:TAffineVectorList);
13394:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
      normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13395:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13396:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13397:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13398:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13399:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13400:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13401:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13402:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13403:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13404:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
      edgesTriangles : TIntegerList) : TIntegerList
13405:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13406:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
      TPersistentObjectList;
13407:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13408:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
      TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13409: end;
13410:
13411: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13412: begin
13413:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13414:  Procedure FreeMemAndNil( var P : TObject)
13415:  Function PCharOrNil( const S : string) : PChar
13416:   SIRegister_TJclReferenceMemoryStream(CL);
13417:   FindClass('TOBJECT'),'EJclVMTError
13418:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13419:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13420:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13421:   PDynamicIndexList', '^TDynamicIndexList // will not work
13422:   PDynamicAddressList', '^TDynamicAddressList // will not work
13423:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13424:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13425:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13426:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13427:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13428:  Function GetInitTable( AClass : TClass) : PTypeInfo
13429:   PFieldEntry', '^TFieldEntry // will not work}
13430:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13431:  Function JIsClass( Address : Pointer) : Boolean
13432:  Function JIsObject( Address : Pointer) : Boolean
13433:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13434:   TDigitCount', 'Integer
13435:   SIRegister_TJclNumericFormat(CL);
13436:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13437:   TTextHandler', 'Procedure ( const Text : string)
13438: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
```

```
13439:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13440:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13441:  Function ReadKey : Char   //to and from the DOS console !
13442:   TModuleHandle', 'HINST
13443:   //TModuleHandle', 'Pointer
13444:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13445:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13446:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13447:  Procedure UnloadModule( var Module : TModuleHandle)
13448:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13449:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13450:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13451:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13452:   FindClass('TOBJECT'),'EJclConversionError
13453:  Function JStrToBoolean( const S : string) : Boolean
13454:  Function JBooleanToStr( B : Boolean) : string
13455:  Function JIntToBool( I : Integer) : Boolean
13456:  Function JBoolToInt( B : Boolean) : Integer
13457:  'ListSeparator','String ';
13458:  'ListSeparator1','String ':
13459:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13460:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13461:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13462:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13463:  Function ListItemCount( const List, Separator : string) : Integer
13464:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13465:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13466:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13467:  Function SystemTObjectInstance : LongWord
13468:  Function IsCompiledWithPackages : Boolean
13469:  Function JJclGUIDToString( const GUID : TGUID) : string
13470:  Function JJclStringToGUID( const S : string) : TGUID
13471:   SIRegister_TJclIntfCriticalSection(CL);
13472:   SIRegister_TJclSimpleLog(CL);
13473:  Procedure InitSimpleLog( const ALogFileName : string)
13474: end;
13475:
13476: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13477: begin
13478:   FindClass('TOBJECT'),'EJclBorRADException
13479:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13480:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13481:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13482:   TJclBorRADToolPath', 'string
13483:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13484:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13485:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13486:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13487:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13488:  BorRADToolRepositoryFormsPage','String 'Forms
13489:  BorRADToolRepositoryProjectsPage','String 'Projects
13490:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13491:  BorRADToolRepositoryObjectType','String 'Type
13492:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13493:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13494:  BorRADToolRepositoryObjectName','String 'Name
13495:  BorRADToolRepositoryObjectPage','String 'Page
13496:  BorRADToolRepositoryObjectIcon','String 'Icon
13497:  BorRADToolRepositoryObjectDescr','String 'Description
13498:  BorRADToolRepositoryObjectAuthor','String 'Author
13499:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13500:  BorRADToolRepositoryObjectDesigner','String 'Designer
13501:  BorRADToolRepositoryDesignerDfm','String 'dfm
13502:  BorRADToolRepositoryDesignerXfm','String 'xfm
13503:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13504:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13505:  SourceExtensionDelphiPackage','String '.dpk
13506:  SourceExtensionBCBPackage','String '.bpk
13507:  SourceExtensionDelphiProject','String '.dpr
13508:  SourceExtensionBCBProject','String '.bpr
13509:  SourceExtensionBDSProject','String '.bdsproj
13510:  SourceExtensionDProject','String '.dproj
13511:  BinaryExtensionPackage','String '.bpl
13512:  BinaryExtensionLibrary','String '.dll
13513:  BinaryExtensionExecutable','String '.exe
13514:  CompilerExtensionDCP','String '.dcp
13515:  CompilerExtensionBPI','String '.bpi
13516:  CompilerExtensionLIB','String '.lib
13517:  CompilerExtensionTDS','String '.tds
13518:  CompilerExtensionMAP','String '.map
13519:  CompilerExtensionDRC','String '.drc
13520:  CompilerExtensionDEF','String '.def
13521:  SourceExtensionCPP','String '.cpp
13522:  SourceExtensionH','String '.h
13523:  SourceExtensionPAS','String '.pas
13524:  SourceExtensionDFM','String '.dfm
13525:  SourceExtensionXFM','String '.xfm
13526:  SourceDescriptionPAS','String 'Pascal source file
```

```
13527:  SourceDescriptionCPP','String 'C++ source file
13528:  DesignerVCL','String 'VCL
13529:  DesignerCLX','String 'CLX
13530:  ProjectTypePackage','String 'package
13531:  ProjectTypeLibrary','String 'library
13532:  ProjectTypeProgram','String 'program
13533:  Personality32Bit','String '32 bit
13534:  Personality64Bit','String '64 bit
13535:  PersonalityDelphi','String 'Delphi
13536:  PersonalityDelphiDotNet','String 'Delphi.net
13537:  PersonalityBCB','String 'C++Builder
13538:  PersonalityCSB','String 'C#Builder
13539:  PersonalityVB','String 'Visual Basic
13540:  PersonalityDesign','String 'Design
13541:  PersonalityUnknown','String 'Unknown personality
13542:  PersonalityBDS','String 'Borland Developer Studio
13543:  DOFDirectoriesSection','String 'Directories
13544:  DOFUnitOutputDirKey','String 'UnitOutputDir
13545:  DOFSearchPathName','String 'SearchPath
13546:  DOFConditionals','String 'Conditionals
13547:  DOFLinkerSection','String 'Linker
13548:  DOFPackagesKey','String 'Packages
13549:  DOFCompilerSection','String 'Compiler
13550:  DOFPackageNoLinkKey','String 'PackageNoLink
13551:  DOFAdditionalSection','String 'Additional
13552:  DOFOptionsKey','String 'Options
13553:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13554:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13555:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13556:   TJclBorPersonalities', 'set of TJclBorPersonality
13557:   TJclBorDesigner', '( bdVCL, bdCLX )
13558:   TJclBorDesigners', 'set of TJClBorDesigner
13559:   TJclBorPlatform', '( bp32bit, bp64bit )
13560:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13561:   SIRegister_TJclBorRADToolInstallationObject(CL);
13562:   SIRegister_TJclBorlandOpenHelp(CL);
13563:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13564:   TJclHelp2Objects', 'set of TJclHelp2Object
13565:   SIRegister_TJclHelp2Manager(CL);
13566:   SIRegister_TJclBorRADToolIdeTool(CL);
13567:   SIRegister_TJclBorRADToolIdePackages(CL);
13568:   SIRegister_IJclCommandLineTool(CL);
13569:   FindClass('TOBJECT'),'EJclCommandLineToolError
13570:   SIRegister_TJclCommandLineTool(CL);
13571:   SIRegister_TJclBorlandCommandLineTool(CL);
13572:   SIRegister_TJclBCC32(CL);
13573:   SIRegister_TJclDCC32(CL);
13574:   TJclDCC', 'TJclDCC32
13575:   SIRegister_TJclBpr2Mak(CL);
13576:   SIRegister_TJclBorlandMake(CL);
13577:   SIRegister_TJclBorRADToolPalette(CL);
13578:   SIRegister_TJclBorRADToolRepository(CL);
13579:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13580:   TCommandLineTools', 'set of TCommandLineTool
13581:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13582:   SIRegister_TJclBorRADToolInstallation(CL);
13583:   SIRegister_TJclBCBInstallation(CL);
13584:   SIRegister_TJclDelphiInstallation(CL);
13585:   SIRegister_TJclDCCIL(CL);
13586:   SIRegister_TJclBDSInstallation(CL);
13587:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13588:   SIRegister_TJclBorRADToolInstallations(CL);
13589:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13590:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13591:  Function IsDelphiPackage( const FileName : string) : Boolean
13592:  Function IsDelphiProject( const FileName : string) : Boolean
13593:  Function IsBCBPackage( const FileName : string) : Boolean
13594:  Function IsBCBProject( const FileName : string) : Boolean
13595:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13596:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13597:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
       Descript:PString;
13598:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString
13599:  function SamePath(const Path1, Path2: string): Boolean;
13600: end;
13601:
13602: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13603: begin
13604:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13605:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13606:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13607:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13608:  'LPathSeparator','String '/
13609:  'LDirDelimiter','String '/
13610:  'LDirSeparator','String ':
13611:  'JXPathDevicePrefix','String '\\.\
13612:  'JXPathSeparator','String '\
13613:  'JXDirDelimiter','String '\
```

```
13614:  'JXDirSeparator','String ';
13615:  'JXPathUncPrefix','String '\\
13616:  'faNormalFile','LongWord')( $00000080);
13617:  //'faUnixSpecific',' faSymLink);
13618:   JXTCompactPath', '( cpCenter, cpEnd )
13619:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13620:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13621:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13622:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13623:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13624:
13625:  Function jxPathAddSeparator( const Path : string) : string
13626:  Function jxPathAddExtension( const Path, Extension : string) : string
13627:  Function jxPathAppend( const Path, Append : string) : string
13628:  Function jxPathBuildRoot( const Drive : Byte) : string
13629:  Function jxPathCanonicalize( const Path : string) : string
13630:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13631:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13632:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13633:  Function jxPathExtractFileDirFixed( const S : string) : string
13634:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13635:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13636:  Function jxPathGetDepth( const Path : string) : Integer
13637:  Function jxPathGetLongName( const Path : string) : string
13638:  Function jxPathGetShortName( const Path : string) : string
13639:  Function jxPathGetLongName( const Path : string) : string
13640:  Function jxPathGetShortName( const Path : string) : string
13641:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13642:  Function jxPathGetTempPath : string
13643:  Function jxPathIsAbsolute( const Path : string) : Boolean
13644:  Function jxPathIsChild( const Path, Base : string) : Boolean
13645:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13646:  Function jxPathIsUNC( const Path : string) : Boolean
13647:  Function jxPathRemoveSeparator( const Path : string) : string
13648:  Function jxPathRemoveExtension( const Path : string) : string
13649:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13650:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13651:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13652:   JxTFileListOptions', 'set of TFileListOption
13653:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13654:   TFileHandler', 'Procedure ( const FileName : string)
13655:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13656:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13657:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
      AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
      FileMatchFunc:TFileMatchFunc):Bool;
13658:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13659:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13660:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13661:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13662:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
      RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13663:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
      IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13664:  Procedure jxCreateEmptyFile( const FileName : string)
13665:  Function jxCloseVolume( var Volume : THandle) : Boolean
13666:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13667:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13668:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13669:  Function jxDelTree( const Path : string) : Boolean
13670:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13671:  Function jxDiskInDrive( Drive : Char) : Boolean
13672:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13673:  Function jxFileCreateTemp( var Prefix : string) : THandle
13674:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13675:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13676:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13677:  Function jxFileExists( const FileName : string) : Boolean
13678:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13679:  Function jxFileRestore( const FileName : string) : Boolean
13680:  Function jxGetBackupFileName( const FileName : string) : string
13681:  Function jxIsBackupFileName( const FileName : string) : Boolean
13682:  Function jxFileGetDisplayName( const FileName : string) : string
13683:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13684:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13685:  Function jxFileGetSize( const FileName : string) : Int64
13686:  Function jxFileGetTempName( const Prefix : string) : string
13687:  Function jxFileGetTypeName( const FileName : string) : string
13688:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13689:  Function jxForceDirectories( Name : string) : Boolean
13690:  Function jxGetDirectorySize( const Path : string) : Int64
13691:  Function jxGetDriveTypeStr( const Drive : Char) : string
13692:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13693:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13694:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13695:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13696:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13697:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
      ResolveSymLinks:Boolean):Integer
```

```
13698:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13699:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13700:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13701:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13702:  Function jxGetFileCreation( const FName : string) : TFileTime;
13703:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13704:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13705:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13706:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13707:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13708:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13709:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13710:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13711:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13712:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13713:  Function jxGetModulePath( const Module : HMODULE) : string
13714:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13715:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13716:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13717:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13718:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13719:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13720:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13721:  Function jxOpenVolume( const Drive : Char) : THandle
13722:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13723:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13724:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13725:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13726:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13727:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13728:  Procedure jxShredFile( const FileName : string; Times : Integer)
13729:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13730:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13731:  Function jxSymbolicLinkTarget( const Name : string) : string
13732:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13733:   SIRegister_TJclCustomFileAttrMask(CL);
13734:   SIRegister_TJclFileAttributeMask(CL);
13735:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13736:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13737:   TFileSearchOptions', 'set of TFileSearchOption
13738:   TFileSearchTaskID', 'Integer
13739:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13740:    +'hTaskID; const Aborted : Boolean)
13741:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13742:   SIRegister_IJclFileEnumerator(CL);
13743:   SIRegister_TJclFileEnumerator(CL);
13744:  Function JxFileSearch : IJclFileEnumerator
13745:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13746:   JxTFileFlags', 'set of TFileFlag
13747:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13748:   SIRegister_TJclFileVersionInfo(CL);
13749:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13750:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13751:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13752:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13753:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13754:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13755:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13756:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13757:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13758:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13759:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13760:   SIRegister_TJclTempFileStream(CL);
13761:   FindClass('TOBJECT'),'TJclCustomFileMapping
13762:   SIRegister_TJclFileMappingView(CL);
13763:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13764:   SIRegister_TJclCustomFileMapping(CL);
13765:   SIRegister_TJclFileMapping(CL);
13766:   SIRegister_TJclSwapFileMapping(CL);
13767:   SIRegister_TJclFileMappingStream(CL);
13768:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13769:   //PPCharArray', '^TPCharArray // will not work
13770:   SIRegister_TJclMappedTextReader(CL);
13771:   SIRegister_TJclFileMaskComparator(CL);
13772:   FindClass('TOBJECT'),'EJclPathError
13773:   FindClass('TOBJECT'),'EJclFileUtilsError
13774:   FindClass('TOBJECT'),'EJclTempFileStreamError
13775:   FindClass('TOBJECT'),'EJclTempFileStreamError
13776:   FindClass('TOBJECT'),'EJclFileMappingError
13777:   FindClass('TOBJECT'),'EJclFileMappingViewError
13778:  Function jxPathGetLongName2( const Path : string) : string
13779:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13780:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13781:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13782:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13783:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13784:  Procedure jxPathListAddItems( var List : string; const Items : string)
```

```
13785:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13786:  Procedure jxPathListDelItems( var List : string; const Items : string)
13787:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13788:  Function jxPathListItemCount( const List : string) : Integer
13789:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13790:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13791:  Function jxPathListItemIndex( const List, Item : string) : Integer
13792:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13793:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13794:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
       AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13795:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
       AllowedPrefixCharacters : string) : Integer
13796: end;
13797:
13798: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13799: begin
13800:  'UTF8FileHeader','String #$ef#$bb#$bf);
13801:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13802:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13803:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13804:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13805:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13806:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13807:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13808:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13809:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13810:  Function lFileIsReadable( const AFilename : string) : boolean
13811:  Function lFileIsWritable( const AFilename : string) : boolean
13812:  Function lFileIsText( const AFilename : string) : boolean
13813:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13814:  Function lFileIsExecutable( const AFilename : string) : boolean
13815:  Function lFileIsSymlink( const AFilename : string) : boolean
13816:  Function lFileIsHardLink( const AFilename : string) : boolean
13817:  Function lFileSize( const Filename : string) : int64;
13818:  Function lGetFileDescription( const AFilename : string) : string
13819:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13820:  Function lTryReadAllLinks( const Filename : string) : string
13821:  Function lDirPathExists( const FileName : String) : Boolean
13822:  Function lForceDirectory( DirectoryName : string) : boolean
13823:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13824:  Function lProgramDirectory : string
13825:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13826:  Function lExtractFileNameOnly( const AFilename : string) : string
13827:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13828:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13829:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13830:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13831:  Function lAppendPathDelim( const Path : string) : string
13832:  Function lChompPathDelim( const Path : string) : string
13833:  Function lTrimFilename( const AFilename : string) : string
13834:  Function lCleanAndExpandFilename( const Filename : string) : string
13835:  Function lCleanAndExpandDirectory( const Filename : string) : string
13836:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13837:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
       AlwaysRequireSharedBaseFolder : Boolean) : string
13838:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13839:  Function lFileIsInPath( const Filename, Path : string) : boolean
13840:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13841:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13842:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13843:  'AllDirectoryEntriesMask','String '*
13844:  Function lGetAllFilesMask : string
13845:  Function lGetExeExt : string
13846:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
       TSearchFileInPathFlags) : string
13847:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
       TSearchFileInPathFlags) : TStrings
13848:  Function lFindDiskFilename( const Filename : string) : string
13849:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13850:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13851:  Function lGetDarwinSystemFilename( Filename : string) : string
13852:   SIRegister_TFileIterator(CL);
13853:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13854:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13855:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13856:   SIRegister_TFileSearcher(CL);
13857:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13858:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13859:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13860:  // TCopyFileFlags', 'set of TCopyFileFlag
13861:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13862:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13863:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13864:  Function lReadFileToString( const Filename : string) : string
13865:  Function lGetTempFilename( const Directory, Prefix : string) : string
13866:  {Function NeedRTLAnsi : boolean
13867:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13868:  Function UTF8ToSys( const s : string) : string
```

```
13869:  Function SysToUTF8( const s : string) : string
13870:  Function ConsoleToUTF8( const s : string) : string
13871:  Function UTF8ToConsole( const s : string) : string}
13872:  Function FileExistsUTF8( const Filename : string) : boolean
13873:  Function FileAgeUTF8( const FileName : string) : Longint
13874:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13875:  Function ExpandFileNameUTF8( const FileName : string) : string
13876:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13877:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13878:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13879:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13880:  Procedure FindCloseUTF8( var F : TSearchrec)
13881:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13882:  Function FileGetAttrUTF8( const FileName : String) : Longint
13883:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13884:  Function DeleteFileUTF8( const FileName : String) : Boolean
13885:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13886:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13887:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13888:  Function GetCurrentDirUTF8 : String
13889:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13890:  Function CreateDirUTF8( const NewDir : String) : Boolean
13891:  Function RemoveDirUTF8( const Dir : String) : Boolean
13892:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13893:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13894:  Function FileCreateUTF8( const FileName : string) : THandle;
13895:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13896:  Function ParamStrUTF8( Param : Integer) : string
13897:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13898:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13899:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13900:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13901:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13902: end;
13903:
13904: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13905: begin
13906:   //VK_F23 = 134;
13907:   //{$EXTERNALSYM VK_F24}
13908:   //VK_F24 = 135;
13909:  TVirtualKeyCode', 'Integer
13910:  'VK_MOUSEWHEELUP','integer'(134);
13911:  'VK_MOUSEWHEELDOWN','integer'(135);
13912:  Function glIsKeyDown( c : Char) : Boolean;
13913:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13914:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13915:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13916:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13917:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13918:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13919: end;
13920:
13921: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13922: begin
13923:   TGLPoint', 'TPoint
13924:   //PGLPoint', '^TGLPoint // will not work
13925:   TGLRect', 'TRect
13926:   //PGLRect', '^TGLRect // will not work
13927:   TDelphiColor', 'TColor
13928:   TGLPicture', 'TPicture
13929:   TGLGraphic', 'TGraphic
13930:   TGLBitmap', 'TBitmap
13931:   //TGraphicClass', 'class of TGraphic
13932:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13933:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
13934:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
13935:    +'Button; Shift : TShiftState; X, Y : Integer)
13936:   TGLMouseMoveEvent', 'TMouseMoveEvent
13937:   TGLKeyEvent', 'TKeyEvent
13938:   TGLKeyPressEvent', 'TKeyPressEvent
13939:   EGLOSError', 'EWin32Error
13940:   EGLOSError', 'EWin32Error
13941:   EGLOSError', 'EOSError
13942:  'glsAllFilter','string'All // sAllFilter
13943:  Function GLPoint( const x, y : Integer) : TGLPoint
13944:  Function GLRGB( const r, g, b : Byte) : TColor
13945:  Function GLColorToRGB( color : TColor) : TColor
13946:  Function GLGetRValue( rgb : DWORD) : Byte
13947:  Function GLGetGValue( rgb : DWORD) : Byte
13948:  Function GLGetBValue( rgb : DWORD) : Byte
13949:  Procedure GLInitWinColors
13950:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13951:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13952:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13953:  Procedure GLInformationDlg( const msg : String)
13954:  Function GLQuestionDlg( const msg : String) : Boolean
13955:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
13956:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
13957:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
```

```
13958:  Function GLApplicationTerminated : Boolean
13959:  Procedure GLRaiseLastOSError
13960:  Procedure GLFreeAndNil( var anObject: TObject)
13961:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
13962:  Function GLGetCurrentColorDepth : Integer
13963:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
13964:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
13965:  Procedure GLSleep( length : Cardinal)
13966:  Procedure GLQueryPerformanceCounter( var val : Int64)
13967:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
13968:  Function GLStartPrecisionTimer : Int64
13969:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
13970:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
13971:  Function GLRDTSC : Int64
13972:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
13973:  Function GLOKMessageBox( const Text, Caption : string) : Integer
13974:  Procedure GLShowHTMLUrl( Url : String)
13975:  Procedure GLShowCursor( AShow : boolean)
13976:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
13977:  Procedure GLGetCursorPos( var point : TGLPoint)
13978:  Function GLGetScreenWidth : integer
13979:  Function GLGetScreenHeight : integer
13980:  Function GLGetTickCount : int64
13981:  function RemoveSpaces(const str : String) : String;
13982:   TNormalMapSpace','( nmsObject, nmsTangent )
13983:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
13984:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
13985:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
13986:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
13987:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
13988: end;
13989:
13990: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
13991: begin
13992:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
13993:  // PGLStarRecord', '^TGLStarRecord // will not work
13994:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
13995:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
13996:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
13997: end;
13998:
13999:
14000: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14001: begin
14002:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14003:   //PAABB', '^TAABB // will not work
14004:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14005:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14006:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14007:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14008:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14009:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14010:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14011:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14012:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14013:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14014:  Function BBMinX( const c : THmgBoundingBox) : Single
14015:  Function BBMaxX( const c : THmgBoundingBox) : Single
14016:  Function BBMinY( const c : THmgBoundingBox) : Single
14017:  Function BBMaxY( const c : THmgBoundingBox) : Single
14018:  Function BBMinZ( const c : THmgBoundingBox) : Single
14019:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14020:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14021:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14022:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14023:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14024:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14025:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14026:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14027:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14028:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14029:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14030:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14031:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14032:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14033:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14034:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14035:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14036:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14037:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14038:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14039:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14040:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14041:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14042:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14043:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
```

```
14044:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14045:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14046:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
        testBSphere:TBSphere):TSpaceContains
14047:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14048:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14049:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14050:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14051:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14052:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
        viewportSizeY:Int):TClipRect
14053: end;
14054:
14055: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14056: begin
14057:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14058:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14059:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14060:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14061:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14062:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14063:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14064:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14065:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14066:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14067:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14068:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14069:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14070:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14071:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14072:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14073:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14074:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14075:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14076:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14077:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14078:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14079:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14080:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14081:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14082: end;
14083:
14084: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14085: begin
14086:  'EPSILON','Single').setExtended( 1e-40);
14087:  'EPSILON2','Single').setExtended( 1e-30);  }
14088:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14089:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14090:  THmgPlane', 'TVector
14091:   TDoubleHmgPlane', 'THomogeneousDblVector
14092:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14093:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14094:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14095:   TSingleArray', 'array of Single
14096:   TTransformations','array [0..15] of Single)
14097:   TPackedRotationMatrix','array [0..2] of Smallint)
14098:   TVertex', 'TAffineVector
14099:   //TVectorGL', 'THomogeneousFltVector
14100:   //TMatrixGL', 'THomogeneousFltMatrix
14101:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14102:  Function glTexPointMake( const s, t : Single) : TTexPoint
14103:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14104:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14105:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14106:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14107:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14108:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14109:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14110:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14111:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14112:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14113:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14114:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14115:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14116:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14117:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14118:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14119:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14120:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14121:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14122:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14123:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14124:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14125:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14126:  Procedure glRstVector( var v : TAffineVector);
14127:  Procedure glRstVector1( var v : TVectorGL);
14128:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14129:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14130:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
```

```
14131:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14132:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14133:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14134:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14135:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14136:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14137:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14138:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14139:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14140:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Int;dest:PTexPointArray);
14141:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14142:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
      PAffineVectorArray);
14143:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14144:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14145:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14146:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14147:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14148:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14149:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14150:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14151:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14152:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14153:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14154:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14155:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14156:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14157:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14158:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14159:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14160:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14161:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14162:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14163:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14164:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14165:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14166:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14167:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14168:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14169:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14170:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14171:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14172:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14173:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14174:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14175:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14176:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14177:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14178:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14179:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14180:  Function glLerp( const start, stop, t : Single) : Single
14181:  Function glAngleLerp( start, stop, t : Single) : Single
14182:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14183:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14184:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14185:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14186:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14187:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14188:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14189:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14190: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14191: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
      PAffineVectorArray);
14192:  Function glVectorLength( const x, y : Single) : Single;
14193:  Function glVectorLength1( const x, y, z : Single) : Single;
14194:  Function glVectorLength2( const v : TAffineVector) : Single;
14195:  Function glVectorLength3( const v : TVectorGL) : Single;
14196:  Function glVectorLength4( const v : array of Single) : Single;
14197:  Function glVectorNorm( const x, y : Single) : Single;
14198:  Function glVectorNorm1( const v : TAffineVector) : Single;
14199:  Function glVectorNorm2( const v : TVectorGL) : Single;
14200:  Function glVectorNorm3( var V : array of Single) : Single;
14201:  Procedure glNormalizeVector( var v : TAffineVector);
14202:  Procedure glNormalizeVector1( var v : TVectorGL);
14203:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14204:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14205: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14206:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14207:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14208:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14209:  Procedure glNegateVector( var V : TAffineVector);
14210:  Procedure glNegateVector2( var V : TVectorGL);
14211:  Procedure glNegateVector3( var V : array of Single);
14212:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14213:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14214:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14215:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
```

```
14216:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14217:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14218:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14219:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14220:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14221:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14222:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14223:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14224:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14225:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14226:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14227:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14228:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14229:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14230:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14231:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14232:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14233:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14234:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14235:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14236:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14237:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14238:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14239:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14240:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14241:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14242:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14243:  Procedure glAbsVector( var v : TVectorGL);
14244:  Procedure glAbsVector1( var v : TAffineVector);
14245:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14246:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14247:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14248:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14249:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14250:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14251:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14252:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14253:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14254:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14255:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14256:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14257:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14258:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14259:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14260:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14261:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14262:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14263:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14264:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14265:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14266:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14267:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14268:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14269:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14270:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14271:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14272:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14273:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14274:  Procedure glAdjointMatrix( var M : TMatrixGL);
14275:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14276:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14277:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14278:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14279:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14280:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14281:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14282:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14283:  Procedure glInvertMatrix( var M : TMatrixGL);
14284:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14285:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14286:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14287:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14288:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14289:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14290:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14291:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14292:  Procedure glNormalizePlane( var plane : THmgPlane)
14293:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14294:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14295:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14296:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14297:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14298:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14299:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14300:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14301:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14302:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14303:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14304:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
```

```
14305:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14306:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14307:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14308:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14309:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14310:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14311:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14312:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14313:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14314:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14315:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14316:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14317:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14318:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14319:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14320:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14321:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14322:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14323:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14324:  Function glLnXP1( X : Extended) : Extended
14325:  Function glLog10( X : Extended) : Extended
14326:  Function glLog2( X : Extended) : Extended;
14327:  Function glLog21( X : Single) : Single;
14328:  Function glLogN( Base, X : Extended) : Extended
14329:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14330:  Function glPower( const Base, Exponent : Single) : Single;
14331:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14332:  Function glDegToRad( const Degrees : Extended) : Extended;
14333:  Function glDegToRad1( const Degrees : Single) : Single;
14334:  Function glRadToDeg( const Radians : Extended) : Extended;
14335:  Function glRadToDeg1( const Radians : Single) : Single;
14336:  Function glNormalizeAngle( angle : Single) : Single
14337:  Function glNormalizeDegAngle( angle : Single) : Single
14338:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14339:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14340:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14341:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14342:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14343:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14344:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14345:  Function glArcCos( const X : Extended) : Extended;
14346:  Function glArcCos1( const x : Single) : Single;
14347:  Function glArcSin( const X : Extended) : Extended;
14348:  Function glArcSin1( const X : Single) : Single;
14349:  Function glArcTan21( const Y, X : Extended) : Extended;
14350:  Function glArcTan21( const Y, X : Single) : Single;
14351:  Function glFastArcTan2( y, x : Single) : Single
14352:  Function glTan( const X : Extended) : Extended;
14353:  Function glTan1( const X : Single) : Single;
14354:  Function glCoTan( const X : Extended) : Extended;
14355:  Function glCoTan1( const X : Single) : Single;
14356:  Function glSinh( const x : Single) : Single;
14357:  Function glSinh1( const x : Double) : Double;
14358:  Function glCosh( const x : Single) : Single;
14359:  Function glCosh1( const x : Double) : Double;
14360:  Function glRSqrt( v : Single) : Single
14361:  Function glRLength( x, y : Single) : Single
14362:  Function glISqrt( i : Integer) : Integer
14363:  Function glILength( x, y : Integer) : Integer;
14364:  Function glILength1( x, y, z : Integer) : Integer;
14365:  Procedure glRegisterBasedExp
14366:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14367:  Function glRoundInt( v : Single) : Single;
14368:  Function glRoundInt1( v : Extended) : Extended;
14369:  Function glTrunc( v : Single) : Integer;
14370:  Function glTrunc64( v : Extended) : Int64;
14371:  Function glInt( v : Single) : Single;
14372:  Function glInt1( v : Extended) : Extended;
14373:  Function glFrac( v : Single) : Single;
14374:  Function glFrac1( v : Extended) : Extended;
14375:  Function glRound( v : Single) : Integer;
14376:  Function glRound64( v : Single) : Int64;
14377:  Function glRound641( v : Extended) : Int64;
14378:  Function glTrunc( X : Extended) : Int64
14379:  Function glRound( X : Extended) : Int64
14380:  Function glFrac( X : Extended) : Extended
14381:  Function glCeil( v : Single) : Integer;
14382:  Function glCeil64( v : Extended) : Int64;
14383:  Function glFloor( v : Single) : Integer;
14384:  Function glFloor64( v : Extended) : Int64;
14385:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14386:  Function glSign( x : Single) : Integer
14387:  Function glIsInRange( const x, a, b : Single) : Boolean;
14388:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14389:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14390:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14391:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14392:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
```

```
14393:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14394:  Function glMinFloat3( const v1, v2 : Single) : Single;
14395:  Function glMinFloat4( const v : array of Single) : Single;
14396:  Function glMinFloat5( const v1, v2 : Double) : Double;
14397:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14398:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14399:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14400:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14401:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14402:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14403:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14404:  Function glMaxFloat2( const v : array of Single) : Single;
14405:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14406:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14407:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14408:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14409:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14410:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14411:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14412:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14413:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14414:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14415:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14416:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14417:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14418:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14419:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14420:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14421:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14422:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14423:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14424:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14425:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14426:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14427:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14428:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14429:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14430:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14431:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14432:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14433:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14434:  Procedure glSortArrayAscending( var a : array of Extended)
14435:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14436:  Function glClampValue1( const aValue, aMin : Single) : Single;
14437:  Function glGeometryOptimizationMode : String
14438:  Procedure glBeginFPUOnlySection
14439:  Procedure glEndFPUOnlySection
14440:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14441:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14442:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14443:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14444:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14445:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14446:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14447:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14448:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14449:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14450:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14451:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14452:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14453:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14454:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14455:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14456:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14457:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14458:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14459:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14460:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14461:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14462:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14463:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14464:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14465:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14466:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14467:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14468:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14469:  'cPI','Single').setExtended( 3.141592654);
14470:  'cPIdiv180','Single').setExtended( 0.017453292);
14471:  'c180divPI','Single').setExtended( 57.29577951);
14472:  'c2PI','Single').setExtended( 6.283185307);
14473:  'cPIdiv2','Single').setExtended( 1.570796326);
14474:  'cPIdiv4','Single').setExtended( 0.785398163);
14475:  'c3PIdiv4','Single').setExtended( 2.35619449);
14476:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
```

```
14477:  'cInv360','Single').setExtended( 1 / 360);
14478:  'c180','Single').setExtended( 180);
14479:  'c360','Single').setExtended( 360);
14480:  'cOneHalf','Single').setExtended( 0.5);
14481:  'cLn10','Single').setExtended( 2.302585093);
14482:  {'MinSingle','Extended').setExtended( 1.5e-45);
14483:  'MaxSingle','Extended').setExtended( 3.4e+38);
14484:  'MinDouble','Extended').setExtended( 5.0e-324);
14485:  'MaxDouble','Extended').setExtended( 1.7e+308);
14486:  'MinExtended','Extended').setExtended( 3.4e-4932);
14487:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14488:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14489:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14490: end;
14491:
14492: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14493: begin
14494:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14495:   (FindClass('TOBJECT'),'TFaceGroups
14496:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14497:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14498:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14499:   SIRegister_TBaseMeshObject(CL);
14500:   (FindClass('TOBJECT'),'TSkeletonFrameList
14501:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14502:   SIRegister_TSkeletonFrame(CL);
14503:   SIRegister_TSkeletonFrameList(CL);
14504:   (FindClass('TOBJECT'),'TSkeleton
14505:   (FindClass('TOBJECT'),'TSkeletonBone
14506:   SIRegister_TSkeletonBoneList(CL);
14507:   SIRegister_TSkeletonRootBoneList(CL);
14508:   SIRegister_TSkeletonBone(CL);
14509:   (FindClass('TOBJECT'),'TSkeletonColliderList
14510:   SIRegister_TSkeletonCollider(CL);
14511:   SIRegister_TSkeletonColliderList(CL);
14512:   (FindClass('TOBJECT'),'TGLBaseMesh
14513:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14514:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14515:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14516:    +'QuaternionList; end
14517:   SIRegister_TSkeleton(CL);
14518:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14519:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14520:   SIRegister_TMeshObject(CL);
14521:   SIRegister_TMeshObjectList(CL);
14522:   //TMeshObjectListClass', 'class of TMeshObjectList
14523:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14524:   SIRegister_TMeshMorphTarget(CL);
14525:   SIRegister_TMeshMorphTargetList(CL);
14526:   SIRegister_TMorphableMeshObject(CL);
14527:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14528:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14529:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14530:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14531:   SIRegister_TSkeletonMeshObject(CL);
14532:   SIRegister_TFaceGroup(CL);
14533:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14534:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14535:   SIRegister_TFGVertexIndexList(CL);
14536:   SIRegister_TFGVertexNormalTexIndexList(CL);
14537:   SIRegister_TFGIndexTexCoordList(CL);
14538:   SIRegister_TFaceGroups(CL);
14539:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14540:   SIRegister_TVectorFile(CL);
14541:   //TVectorFileClass', 'class of TVectorFile
14542:   SIRegister_TGLGLSMVectorFile(CL);
14543:   SIRegister_TGLBaseMesh(CL);
14544:   SIRegister_TGLFreeForm(CL);
14545:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14546:   TGLActorOptions', 'set of TGLActorOption
14547:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14548:   (FindClass('TOBJECT'),'TGLActor
14549:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14550:   SIRegister_TActorAnimation(CL);
14551:   TActorAnimationName', 'String
14552:   SIRegister_TActorAnimations(CL);
14553:   SIRegister_TGLBaseAnimationControler(CL);
14554:   SIRegister_TGLAnimationControler(CL);
14555:   TActorFrameInterpolation', '( afpNone, afpLinear )
14556:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14557:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14558:   SIRegister_TGLActor(CL);
14559:   SIRegister_TVectorFileFormat(CL);
14560:   SIRegister_TVectorFileFormatsList(CL);
14561:   (FindClass('TOBJECT'),'EInvalidVectorFile
14562:  Function GetVectorFileFormats : TVectorFileFormatsList
14563:  Function VectorFileFormatsFilter : String
14564:  Function VectorFileFormatsSaveFilter : String
14565:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
```

```
14566:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14567:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14568: end;
14569:
14570: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14571: begin
14572:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14573:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14574:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14575:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14576:   SIRegister_TOleStream(CL);
14577:   (FindClass('TOBJECT'),'TConnectionPoints
14578:   TConnectionKind', '( ckSingle, ckMulti )
14579:   SIRegister_TConnectionPoint(CL);
14580:   SIRegister_TConnectionPoints(CL);
14581:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14582:   (FindClass('TOBJECT'),'TActiveXControlFactory
14583:   SIRegister_TActiveXControl(CL);
14584:   //TActiveXControlClass', 'class of TActiveXControl
14585:   SIRegister_TActiveXControlFactory(CL);
14586:   SIRegister_TActiveFormControl(CL);
14587:   SIRegister_TActiveForm(CL);
14588:   //TActiveFormClass', 'class of TActiveForm
14589:   SIRegister_TActiveFormFactory(CL);
14590:   (FindClass('TOBJECT'),'TPropertyPageImpl
14591:   SIRegister_TPropertyPage(CL);
14592:   //TPropertyPageClass', 'class of TPropertyPage
14593:   SIRegister_TPropertyPageImpl(CL);
14594:   SIRegister_TActiveXPropertyPage(CL);
14595:   SIRegister_TActiveXPropertyPageFactory(CL);
14596:   SIRegister_TCustomAdapter(CL);
14597:   SIRegister_TAdapterNotifier(CL);
14598:   SIRegister_IFontAccess(CL);
14599:   SIRegister_TFontAdapter(CL);
14600:   SIRegister_IPictureAccess(CL);
14601:   SIRegister_TPictureAdapter(CL);
14602:   SIRegister_TOleGraphic(CL);
14603:   SIRegister_TStringsAdapter(CL);
14604:   SIRegister_TReflectorWindow(CL);
14605:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14606:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14607:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14608:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14609:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14610:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14611:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14612:  Function ParkingWindow : HWND
14613: end;
14614:
14615: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14616: begin
14617:  // TIp6Bytes = array [0..15] of Byte;
14618: {:binary form of IPv6 adress (for string conversion routines)}
14619:  // TIp6Words = array [0..7] of Word;
14620:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14621:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14622:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14623:  Function synaIsIP( const Value : string) : Boolean');
14624:  Function synaIsIP6( const Value : string) : Boolean');
14625:  Function synaIPToID( Host : string) : Ansistring');
14626:  Function synaStrToIp6( value : string) : TIp6Bytes');
14627:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14628:  Function synaStrToIp( value : string) : integer');
14629:  Function synaIpToStr( value : integer) : string');
14630:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14631:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14632:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14633:  Function xStrToIP( const Value : String) : TIPAdr');
14634:  Function xIPToStr( const Adresse : TIPAdr) : String');
14635:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14636:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14637: end;
14638:
14639: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14640: begin
14641:  AddTypeS('TSpecials', 'set of Char');
14642:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14643:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14644:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14645:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14646:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14647:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14648:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14649:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14650:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14651:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14652:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14653:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14654:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
```

```
14655:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14656:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14657:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14658:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
14659:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14660:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14661:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14662:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14663:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14664:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14665:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14666:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14667:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14668:  Function synCrc32( const Value : AnsiString) : Integer');
14669:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14670:  Function Crc16( const Value : AnsiString) : Word');
14671:  Function synMD5( const Value : AnsiString) : AnsiString');
14672:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14673:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14674:  Function synSHA1( const Value : AnsiString) : AnsiString');
14675:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14676:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14677:  Function synMD4( const Value : AnsiString) : AnsiString');
14678: end;
14679:
14680: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14681: begin
14682:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14683:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14684:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14685:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14686:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14687:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14688:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14689:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14690:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14691:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14692:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14693:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14694:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14695:    +', CP864, CP865, CP869, CP1125 )');
14696:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14697:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14698:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
       TransformTable : array of Word) : AnsiString');
14699:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
       TransformTable : array of Word; Translit : Boolean) : AnsiString');
14700:  Function GetCurCP : TMimeChar');
14701:  Function GetCurOEMCP : TMimeChar');
14702:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14703:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14704:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14705:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14706:  Function GetBOM( Value : TMimeChar) : AnsiString');
14707:  Function StringToWide( const Value : AnsiString) : WideString');
14708:  Function WideToString( const Value : WideString) : AnsiString');
14709: end;
14710:
14711: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14712: begin
14713:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14714:  Procedure WakeOnLan( MAC, IP : string)');
14715:  Function GetDNS : string');
14716:  Function GetIEProxy( protocol : string) : TProxySetting');
14717:  Function GetLocalIPs : string');
14718: end;
14719:
14720:
14721: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14722: begin
14723:  AddConstantN('synCR','Char #$0d);
14724:  Const('synLF','Char #$0a);
14725:  Const('cSerialChunk','LongInt'( 8192);
14726:  Const('LockfileDirectory','String '/var/lock');
14727:  Const('PortIsClosed','LongInt'( - 1);
14728:  Const('ErrAlreadyOwned','LongInt'( 9991);
14729:  Const('ErrAlreadyInUse','LongInt'( 9992);
14730:  Const('ErrWrongParameter','LongInt'( 9993);
14731:  Const('ErrPortNotOpen','LongInt'( 9994);
14732:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14733:  Const('ErrMaxBuffer','LongInt'( 9996);
14734:  Const('ErrTimeout','LongInt'( 9997);
14735:  Const('ErrNotRead','LongInt'( 9998);
14736:  Const('ErrFrame','LongInt'( 9999);
14737:  Const('ErrOverrun','LongInt'( 10000);
14738:  Const('ErrRxOver','LongInt'( 10001);
14739:  Const('ErrRxParity','LongInt'( 10002);
14740:  Const('ErrTxFull','LongInt'( 10003);
14741:  Const('dcb_Binary','LongWord')( $00000001);
```

```
14742:  Const('dcb_ParityCheck','LongWord')( $00000002);
14743:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14744:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14745:  Const('dcb_DtrControlMask','LongWord')( $00000030);
14746:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14747:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14748:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14749:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14750:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14751:  Const('dcb_OutX','LongWord')( $00000100);
14752:  Const('dcb_InX','LongWord')( $00000200);
14753:  Const('dcb_ErrorChar','LongWord')( $00000400);
14754:  Const('dcb_NullStrip','LongWord')( $00000800);
14755:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14756:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14757:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14758:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14759:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14760:  Const('dcb_AbortOnError','LongWord')( $00004000);
14761:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14762:  Const('synSB1','LongInt'( 0);
14763:  Const('SB1andHalf','LongInt'( 1);
14764:  Const('synSB2','LongInt'( 2);
14765:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14766:  Const('CS7fix','LongWord')( $0000020);
14767:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14768:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14769:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14770:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14771:   //AddTypeS('PDCB', '^TDCB // will not work');
14772:  //Const('MaxRates','LongInt'( 18);
14773:  //Const('MaxRates','LongInt'( 30);
14774:  //Const('MaxRates','LongInt'( 19);
14775:  Const('O_SYNC','LongWord')( $0080);
14776:  Const('synOK','LongInt'( 0);
14777:  Const('synErr','LongInt'( integer ( - 1 ));
14778:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14779:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14780:   SIRegister_ESynaSerError(CL);
14781:   SIRegister_TBlockSerial(CL);
14782:  Function GetSerialPortNames : string');
14783: end;
14784:
14785: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14786: begin
14787:  Const('DLLIconvName','String 'libiconv.so');
14788:  Const('DLLIconvName','String 'iconv.dll');
14789:   AddTypeS('size_t', 'Cardinal');
14790:   AddTypeS('iconv_t', 'Integer');
14791:   //AddTypeS('iconv_t', 'Pointer');
14792:   AddTypeS('argptr', 'iconv_t');
14793:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14794:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14795:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14796:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14797:  Function SynaIconvClose( var cd : iconv_t) : integer');
14798:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14799:  Function IsIconvloaded : Boolean');
14800:  Function InitIconvInterface : Boolean');
14801:  Function DestroyIconvInterface : Boolean');
14802:  Const('ICONV_TRIVIALP','LongInt'( 0);
14803:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14804:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14805:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14806:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14807: end;
14808:
14809: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14810: begin
14811:  Const 'ICMP_ECHO','LongInt'( 8);
14812:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14813:  Const('ICMP_UNREACH','LongInt'( 3);
14814:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14815:  Const('ICMP6_ECHO','LongInt'( 128);
14816:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14817:  Const('ICMP6_UNREACH','LongInt'( 1);
14818:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14819:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14820:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14821:   SIRegister_TPINGSend(CL);
14822:  Function PingHost( const Host : string) : Integer');
14823:  Function TraceRouteHost( const Host : string) : string');
14824: end;
14825:
14826: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14827: begin
14828:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14829:  Const('synASN1_INT','LongWord')( $02);
```

```
14830:  Const('synASN1_OCTSTR','LongWord')( $04);
14831:  Const('synASN1_NULL','LongWord')( $05);
14832:  Const('synASN1_OBJID','LongWord')( $06);
14833:  Const('synASN1_ENUM','LongWord')( $0a);
14834:  Const('synASN1_SEQ','LongWord')( $30);
14835:  Const('synASN1_SETOF','LongWord')( $31);
14836:  Const('synASN1_IPADDR','LongWord')( $40);
14837:  Const('synASN1_COUNTER','LongWord')( $41);
14838:  Const('synASN1_GAUGE','LongWord')( $42);
14839:  Const('synASN1_TIMETICKS','LongWord')( $43);
14840:  Const('synASN1_OPAQUE','LongWord')( $44);
14841:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14842:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14843:  Function synASNEncLen( Len : Integer) : AnsiString');
14844:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14845:  Function synASNEncInt( Value : Integer) : AnsiString');
14846:  Function synASNEncUInt( Value : Integer) : AnsiString');
14847:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14848:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14849:  Function synMibToId( Mib : String) : AnsiString');
14850:  Function synIdToMib( const Id : AnsiString) : String');
14851:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14852:  Function ASNdump( const Value : AnsiString) : AnsiString');
14853:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14854:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14855: end;
14856:
14857: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14858: begin
14859:  Const('cLDAPProtocol','String '389');
14860:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14861:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14862:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14863:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14864:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14865:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14866:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14867:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14868:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14869:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14870:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14871:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14872:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14873:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14874:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14875:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14876:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14877:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14878:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14879:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14880:   SIRegister_TLDAPAttribute(CL);
14881:   SIRegister_TLDAPAttributeList(CL);
14882:   SIRegister_TLDAPResult(CL);
14883:   SIRegister_TLDAPResultList(CL);
14884:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14885:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14886:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14887:   SIRegister_TLDAPSend(CL);
14888:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14889: end;
14890:
14891:
14892: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14893: begin
14894:  Const('cSysLogProtocol','String '514');
14895:  Const('FCL_Kernel','LongInt'( 0);
14896:  Const('FCL_UserLevel','LongInt'( 1);
14897:  Const('FCL_MailSystem','LongInt'( 2);
14898:  Const('FCL_System','LongInt'( 3);
14899:  Const('FCL_Security','LongInt'( 4);
14900:  Const('FCL_Syslogd','LongInt'( 5);
14901:  Const('FCL_Printer','LongInt'( 6);
14902:  Const('FCL_News','LongInt'( 7);
14903:  Const('FCL_UUCP','LongInt'( 8);
14904:  Const('FCL_Clock','LongInt'( 9);
14905:  Const('FCL_Authorization','LongInt'( 10);
14906:  Const('FCL_FTP','LongInt'( 11);
14907:  Const('FCL_NTP','LongInt'( 12);
14908:  Const('FCL_LogAudit','LongInt'( 13);
14909:  Const('FCL_LogAlert','LongInt'( 14);
14910:  Const('FCL_Time','LongInt'( 15);
14911:  Const('FCL_Local0','LongInt'( 16);
14912:  Const('FCL_Local1','LongInt'( 17);
14913:  Const('FCL_Local2','LongInt'( 18);
14914:  Const('FCL_Local3','LongInt'( 19);
14915:  Const('FCL_Local4','LongInt'( 20);
14916:  Const('FCL_Local5','LongInt'( 21);
14917:  Const('FCL_Local6','LongInt'( 22);
14918:  Const('FCL_Local7','LongInt'( 23);
```

```
14919:   Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
14920:   SIRegister_TSyslogMessage(CL);
14921:   SIRegister_TSyslogSend(CL);
14922:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
        Content:string):Boolean;
14923: end;
14924:
14925:
14926: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14927: begin
14928:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14929:   SIRegister_TMessHeader(CL);
14930:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14931:   SIRegister_TMimeMess(CL);
14932: end;
14933:
14934: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14935: begin
14936:   (FindClass('TOBJECT'),'TMimePart');
14937:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
14938:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14939:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14940:   SIRegister_TMimePart(CL);
14941:  Const('MaxMimeType','LongInt'( 25);
14942:  Function GenerateBoundary : string');
14943: end;
14944:
14945: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14946: begin
14947:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
14948:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
14949:  Function NeedInline( const Value : AnsiString) : boolean');
14950:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
14951:  Function InlineCode( const Value : string) : string');
14952:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
14953:  Function InlineEmail( const Value : string) : string');
14954: end;
14955:
14956: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
14957: begin
14958:  Const('cFtpProtocol','String '21');
14959:  Const('cFtpDataProtocol','String '20');
14960:  Const('FTP_OK','LongInt'( 255);
14961:  Const('FTP_ERR','LongInt'( 254);
14962:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
14963:   SIRegister_TFTPListRec(CL);
14964:   SIRegister_TFTPList(CL);
14965:   SIRegister_TFTPSend(CL);
14966:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
14967:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
14968:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
        ToPort, ToFile, ToUser, ToPass : string) : Boolean');
14969: end;
14970:
14971: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
14972: begin
14973:  Const('cHttpProtocol','String '80');
14974:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
14975:   SIRegister_THTTPSend(CL);
14976:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
14977:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
14978:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
14979:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
14980:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
        ResultData:TStrings):Bool;
14981: end;
14982:
14983: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
14984: begin
14985:  Const('cSmtpProtocol','String '25');
14986:   SIRegister_TSMTPSend(CL);
14987:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
        Passw:string):Bool;
14988:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
14989:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
        Username, Password : string):Boolean');
14990: end;
14991:
14992: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
14993: begin
14994:  Const('cSnmpProtocol','String '161');
14995:  Const('cSnmpTrapProtocol','String '162');
14996:  Const('SNMP_V1','LongInt'( 0);
14997:  Const('SNMP_V2C','LongInt'( 1);
14998:  Const('SNMP_V3','LongInt'( 3);
14999:  Const('PDUGetRequest','LongWord')( $A0);
15000:  Const('PDUGetNextRequest','LongWord')( $A1);
15001:  Const('PDUGetResponse','LongWord')( $A2);
15002:  Const('PDUSetRequest','LongWord')( $A3);
```

```
15003:  Const('PDUTrap','LongWord')( $A4);
15004:  Const('PDUGetBulkRequest','LongWord')( $A5);
15005:  Const('PDUInformRequest','LongWord')( $A6);
15006:  Const('PDUTrapV2','LongWord')( $A7);
15007:  Const('PDUReport','LongWord')( $A8);
15008:  Const('ENoError',LongInt 0;
15009:  Const('ETooBig','LongInt')( 1);
15010:  Const('ENoSuchName','LongInt'( 2);
15011:  Const('EBadValue','LongInt'( 3);
15012:  Const('EReadOnly','LongInt'( 4);
15013:  Const('EGenErr','LongInt'( 5);
15014:  Const('ENoAccess','LongInt'( 6);
15015:  Const('EWrongType','LongInt'( 7);
15016:  Const('EWrongLength','LongInt'( 8);
15017:  Const('EWrongEncoding','LongInt'( 9);
15018:  Const('EWrongValue','LongInt'( 10);
15019:  Const('ENoCreation','LongInt'( 11);
15020:  Const('EInconsistentValue','LongInt'( 12);
15021:  Const('EResourceUnavailable','LongInt'( 13);
15022:  Const('ECommitFailed','LongInt'( 14);
15023:  Const('EUndoFailed','LongInt'( 15);
15024:  Const('EAuthorizationError','LongInt'( 16);
15025:  Const('ENotWritable','LongInt'( 17);
15026:  Const('EInconsistentName','LongInt'( 18);
15027:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15028:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15029:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15030:   SIRegister_TSNMPMib(CL);
15031:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15032:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15033:   SIRegister_TSNMPRec(CL);
15034:   SIRegister_TSNMPSend(CL);
15035:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15036:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15037:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15038:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15039:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15040:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15041:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : TStringList) : Integer');
15042: end;
15043:
15044: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15045: begin
15046:  Function GetDomainName2: AnsiString');
15047:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15048:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15049:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15050:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15051:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15052: end;
15053:
15054: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15055: begin
15056:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15057:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15058:  Function wwStrToDate( const S : string) : boolean');
15059:  Function wwStrToTime( const S : string) : boolean');
15060:  Function wwStrToDateTime( const S : string) : boolean');
15061:  Function wwStrToTimeVal( const S : string) : TDateTime');
15062:  Function wwStrToDateVal( const S : string) : TDateTime');
15063:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15064:  Function wwStrToInt( const S : string) : boolean');
15065:  Function wwStrToFloat( const S : string) : boolean');
15066:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15067:  Function wwNextDay( Year, Month, Day : Word) : integer');
15068:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15069:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15070:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15071:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15072:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15073:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15074:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15075:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15076:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15077: end;
15078:
15079: unit uPSI_Themes;
15080: Function ThemeServices : TThemeServices');
15081: Function ThemeControl( AControl : TControl) : Boolean');
15082: Function UnthemedDesigner( AControl : TControl) : Boolean');
15083: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15084: begin
15085:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15086: end;
15087: Unit uPSC_menus;
15088:  Function StripHotkey( const Text : string) : string');
15089:  Function GetHotkey( const Text : string) : string');
```

```
15090:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15091:  Function IsAltGRPressed : boolean');
15092:
15093: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15094: begin
15095:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15096:   SIRegister_TIdIMAP4Server(CL);
15097: end;
15098:
15099: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15100: begin
15101:  'HASH_SIZE','LongInt'( 256);
15102:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15103:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15104:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15105:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15106:    +' : Integer; Value : Variant; end');
15107:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15108:   SIRegister_TVariantSymbolTable(CL);
15109: end;
15110:
15111: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15112: begin
15113:   SIRegister_TThreadLocalVariables(CL);
15114:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15115:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15116:  Function ThreadLocals : TThreadLocalVariables');
15117:  Procedure WriteDebug( sz : String)');
15118:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15119:  'UDF_FAILURE','LongInt'( 1);
15120:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15121:  CL.AddTypeS('mTByteArray', 'array of byte;');
15122:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15123:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15124:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15125:  function IsNetworkConnected: Boolean;
15126:  function IsInternetConnected: Boolean;
15127:  function IsCOMConnected: Boolean;
15128:  function IsNetworkOn: Boolean;
15129:  function IsInternetOn: Boolean;
15130:  function IsCOMOn: Boolean;
15131:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15132:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15133:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15134:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15135:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15136:  Function GetMenu( hWnd : HWND) : HMENU');
15137:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15138: end;
15139:
15140: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15141: begin
15142:   SIRegister_IDataBlock(CL);
15143:   SIRegister_ISendDataBlock(CL);
15144:   SIRegister_ITransport(CL);
15145:   SIRegister_TDataBlock(CL);
15146:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15147:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15148:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15149:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15150:   SIRegister_TCustomDataBlockInterpreter(CL);
15151:   SIRegister_TSendDataBlock(CL);
15152:   'CallSig','LongWord')( $D800);
15153:   'ResultSig','LongWord')( $D400);
15154:   'asMask','LongWord')( $00FF);
15155:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15156:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15157:  Procedure CheckSignature( Sig : Integer)');
15158: end;
15159:
15160: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15161: begin
15162:   //CL.AddTypeS('HINTERNET', '___Pointer');
15163:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15164:   CL.AddTypeS('HINTERNET', 'Integer');
15165:   CL.AddTypeS('HINTERNET2', '___Pointer');
15166:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15167:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15168:   CL.AddTypeS('INTERNET_PORT', 'Word');
15169:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15170:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15171:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15172:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15173:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15174:  Function InternetCrackUrl(lpszUrl:PChar; dwUrlLength,dwFlags:DWORD; var
       lpUrlComponents:TURLComponents):BOOL;
15175:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
       dwUrlLength:DWORD):BOOL;
15176:  Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
```

```
15177:  Function
        InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
        lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15178:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15179:  ;dwContext:DWORD):HINTERNET;
15180:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
        lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15181:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
        dwContext:DWORD):BOOL;
15182:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15183:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15184:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15185:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15186:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15187:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15188:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15189:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
        lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15190:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
        lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15191:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15192:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15193:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
        dwContext:DWORD):BOOL;
15194:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
        TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15195:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
        BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15196:  Function
        WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15197:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15198:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15199:  Function
        FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15200:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15201:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15202:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15203:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15204:  Function
        FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15205:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15206:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15207:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15208:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15209:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15210:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15211:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15212:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15213:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15214:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15215:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15216:  Function
        GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
        PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15217:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15218:  Function
        GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15219:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
        PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15220:  Function
        HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15221:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
        dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15222:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15223:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15224:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15225:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15226:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15227:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15228:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15229:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
        lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15230:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
        lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15231:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15232:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15233:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15234:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15235: end;
15236:
15237: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15238: begin
15239:   AddTypeS('strCharSet', 'set of char');
15240:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15241:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15242:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15243:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
```

```
15244:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15245:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15246:  Procedure strStripWhiteSpace( var s : string)');
15247:  Function strRemoveChar( str : string; removeChar : char) : string');
15248:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15249:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15250:  Function wwEqualStr( s1, s2 : string) : boolean');
15251:  Function strCount( s : string; delimeter : char) : integer');
15252:  Function strWhiteSpace : strCharSet');
15253:  Function wwExtractFileNameOnly( const FileName : string) : string');
15254:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15255:  Function strTrailing( s : string; delimeter : char) : string');
15256:  Function strPreceding( s : string; delimeter : char) : string');
15257:  Function wwstrReplace( s, Find, Replace : string) : string');
15258: end;
15259:
15260: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15261: begin
15262:   SIRegister_TRemoteDataModule(CL);
15263:   SIRegister_TCRemoteDataModule(CL);
15264:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15265:  Procedure UnregisterPooled( const ClassID : string)');
15266:  Procedure EnableSocketTransport( const ClassID : string)');
15267:  Procedure DisableSocketTransport( const ClassID : string)');
15268:  Procedure EnableWebTransport( const ClassID : string)');
15269:  Procedure DisableWebTransport( const ClassID : string)');
15270: end;
15271:
15272: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15273: begin
15274:  Function mxArcCos( x : Real) : Real');
15275:  Function mxArcSin( x : Real) : Real');
15276:  Function Comp2Str( N : Comp) : String');
15277:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15278:  Function Int2Str( N : LongInt) : String');
15279:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15280:  Function LogXY( x, y : Real) : Real');
15281:  Function Pennies2Dollars( C : Comp) : String');
15282:  Function mxPower( X : Integer; Y : Integer) : Real');
15283:  Function Real2Str( N : Real; Width, Places : integer) : String');
15284:  Function mxStr2Comp( MyString : string) : Comp');
15285:  Function mxStr2Pennies( S : String) : Comp');
15286:  Function Str2Real( MyString : string) : Real');
15287:  Function XToTheY( x, y : Real) : Real');
15288: end;
15289:
15290: //************************Cindy Functions!*****************************
15291: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15292: begin
15293:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15294:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15295:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15296:  MessagePlainText','String 'text/plain');
15297:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15298:  MessageAlterText_Html','String 'multipart/alternative');
15299:  MessageHtml_Attach','String 'multipart/mixed');
15300:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15301:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15302:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15303:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15304:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15305:  Function ForceDecodeHeader( aHeader : String) : String');
15306:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15307:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15308:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15309:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15310:  Function Get_MD5( const aFileName : string) : string');
15311:  Function Get_MD5FromString( const aString : string) : string');
15312: end;
15313:
15314: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15315: begin
15316:  Function IsFolder( SRec : TSearchrec) : Boolean');
15317:  Function isFolderReadOnly( Directory : String) : Boolean');
15318:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15319:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15320:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15321:  Function DiskFreeBytes( Drv : Char) : Int64');
15322:  Function DiskBytes( Drv : Char) : Int64');
15323:  Function GetFileBytes( Filename : String) : Int64');
15324:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15325:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15326:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15327:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15328:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15329:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15330:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15331:  SE_TCB_NAME','String 'SeTcbPrivilege');
15332:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
```

```
15333:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15334:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15335:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15336:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15337:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15338:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15339:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15340:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15341:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15342:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15343:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15344:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
15345:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15346:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15347:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15348:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15349:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15350:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15351:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15352:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15353: end;
15354:
15355:
15356: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15357: begin
15358:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15359:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15360:  Function ShellGetExtensionName( FileName : String) : String');
15361:  Function ShellGetIconIndex( FileName : String) : Integer');
15362:  Function ShellGetIconHandle( FileName : String) : HIcon');
15363:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15364:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15365:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15366:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15367:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15368:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15369:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15370:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
       WaitCloseCompletion : boolean) : Boolean');
15371:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15372:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15373:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15374:  Function GetModificationDate( Filename : String) : TDateTime');
15375:  Function GetCreationDate( Filename : String) : TDateTime');
15376:  Function GetLastAccessDate( Filename : String) : TDateTime');
15377:  Function FileDelete( Filename : String) : Boolean');
15378:  Function FileIsOpen( Filename : string) : boolean');
15379:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15380:  Function DirectoryDelete( Directory : String) : Boolean');
15381:  Function GetPrinters( PrintersList : TStrings) : Integer');
15382:  Procedure SetDefaultPrinter( PrinterName : String)');
15383:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15384:  Function WinToDosPath( WinPathName : String) : String');
15385:  Function DosToWinPath( DosPathName : String) : String');
15386:  Function cyGetWindowsVersion : TWindowsVersion');
15387:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15388:  Procedure WindowsShutDown( Restart : boolean)');
15389:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
       FileIcon:string;NumIcone:integer);
15390:  Procedure GetWindowsFonts( FontsList : TStrings)');
15391:  Function GetAvailableFilename( DesiredFileName : String): String');
15392: end;
15393:
15394: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15395: begin
15396:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15397:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15398:   Type(TStringRead', '( srFromLeft, srFromRight )');
15399:   Type(TStringReads', 'set of TStringRead');
15400:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15401:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15402:   Type(TWordsOptions', 'set of TWordsOption');
15403:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15404:   Type(TCarTypes', 'set of TCarType');
15405:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15406:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15407:  Function Char_GetType( aChar : Char) : TCarType');
15408:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15409:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15410:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15411:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15412:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15413:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15414:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15415:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15416:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15417:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15418:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15419:  Function String_Quote( Str : String) : String');
```

```
15420:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15421:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15422:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15423:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15424:  Function String_ToInt( Str : String) : Integer');
15425:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15426:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15427:  Function String_Reverse( Str : String) : String');
15428:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15429:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15430:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15431:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15432:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
       Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15433:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15434:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15435:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15436:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15437:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15438:  Function String_End( Str : String; Cars : Word) : String');
15439:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15440:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15441:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15442:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15443:  Function String_IsNumbers( Str : String) : Boolean');
15444:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15445:  Function StringToCsvCell( aStr : String) : String');
15446: end;
15447:
15448: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15449: begin
15450:  Function LongDayName( aDate : TDate) : String');
15451:  Function LongMonthName( aDate : TDate) : String');
15452:  Function ShortYearOf( aDate : TDate) : byte');
15453:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15454:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15455:  Function SecondsToMinutes( Seconds : Integer) : Double');
15456:  Function MinutesToSeconds( Minutes : Double) : Integer');
15457:  Function MinutesToHours( Minutes : Integer) : Double');
15458:  Function HoursToMinutes( Hours : Double) : Integer');
15459:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15460:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15461:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15462:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15463:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15464:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15465:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15466:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15467:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15468: end;
15469:
15470: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15471: begin
15472:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15473:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15474:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15475:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15476:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15477:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15478:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15479:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15480:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15481:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15482:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15483:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15484:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15485:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
15486:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15487:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15488:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15489:  Procedure cyCenterControl( aControl : TControl)');
15490:  Function GetLastParent( aControl : TControl) : TWinControl');
15491:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15492:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15493: end;
15494:
15495: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15496: begin
15497:  Function TablePackTable( Tab : TTable) : Boolean');
15498:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15499:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15500:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15501:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15502:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
```

```
15503:  Function TableEmptyTable( Tab : TTable) : Boolean');
15504:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15505:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15506:  Function
        TableCreate(Owner:TComponent;DataBaseName:ShortString;TableName:String;IndexName:ShortString;ReadOnly :
        Boolean):TTable;
15507:  Function
        TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15508:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15509: end;
15510:
15511: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15512: begin
15513:   SIRegister_TcyRunTimeDesign(CL);
15514:   SIRegister_TcyShadowText(CL);
15515:   SIRegister_TcyBgPicture(CL);
15516:   SIRegister_TcyGradient(CL);
15517:   SIRegister_tcyBevel(CL);
15518:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15519:   SIRegister_tcyBevels(CL);
15520:   SIRegister_TcyImagelistOptions(CL);
15521:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15522: end;
15523:
15524: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15525: begin
15526:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
        adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
        Maxdegrade : Byte;'+
15527:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15528:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15529:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15530:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
        Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15531:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
        Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15532:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15533:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15534:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
        BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15535:  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;
15536:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
        aState : TButtonState; Focused, Hot : Boolean)');
15537:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
        aState : TButtonState; Focused, Hot : Boolean)');
15538:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
        TColor; aState : TButtonState; Focused, Hot : Boolean)');
15539:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
        GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15540:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
        const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15541:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
        TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15542:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
        TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15543:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
        WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15544:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15545:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15546:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15547:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
        CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15548:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15549:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15550:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15551:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15552:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15553:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15554:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15555:  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);
15556:  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;
15557:  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);');
15558:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15559:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15560:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15561:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15562:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
```

```
15563:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15564:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15565:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15566:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15567:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15568:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15569:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15570:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15571:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15572:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15573:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15574: end;
15575:
15576: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15577: begin
15578:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15579:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15580:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15581:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15582:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15583:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15584:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15585:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15586:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15587:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15588:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15589:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
       bmInvertReverseFromColor);
15590:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
       rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15591:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15592:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15593: end;
15594:
15595: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15596: begin
15597:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15598:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15599:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15600:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15601:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15602:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15603:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15604:  Const SC_GROUP_IDENTIFIERW','String') '+');
15605:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15606:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15607:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15608:  Const SERVICE_INACTIVE','LongWord $00000002);
15609:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15610:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15611:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15612:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15613:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15614:  Const SERVICE_STOPPED','LongWord $00000001);
15615:  Const SERVICE_START_PENDING','LongWord $00000002);
15616:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15617:  Const SERVICE_RUNNING','LongWord $00000004);
15618:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15619:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15620:  Const SERVICE_PAUSED','LongWord $00000007);
15621:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15622:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15623:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15624:  Const SC_MANAGER_CONNECT','LongWord $0001);
15625:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15626:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15627:  Const SC_MANAGER_LOCK','LongWord $0008);
15628:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15629:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15630:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15631:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15632:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15633:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15634:  Const SERVICE_START','LongWord $0010);
15635:  Const SERVICE_STOP','LongWord $0020);
15636:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15637:  Const SERVICE_INTERROGATE','LongWord $0080);
15638:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15639:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15640:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15641:  Const SERVICE_ADAPTER','LongWord $00000004);
15642:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15643:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15644:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15645:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15646:  Const SERVICE_BOOT_START','LongWord $00000000);
15647:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15648:  Const SERVICE_AUTO_START','LongWord $00000002);
15649:  Const SERVICE_DEMAND_START','LongWord $00000003);
```

```
15650:  Const SERVICE_DISABLED','LongWord $00000004);
15651:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15652:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15653:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15654:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15655:   CL.AddTypeS('SC_HANDLE', 'THandle');
15656:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15657:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15658:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15659:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15660:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15661:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15662:   Const TServiceStatus', '_SERVICE_STATUS');
15663:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15664:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15665:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15666:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15667:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15668:   TEnumServiceStatus', 'TEnumServiceStatusA');
15669:   SC_LOCK', '___Pointer');
15670:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15671:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15672:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15673:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15674:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15675:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15676:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15677:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15678:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15679:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15680:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15681:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15682:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15683:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15684:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15685:   TQueryServiceConfig', 'TQueryServiceConfigA');
15686:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15687:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15688:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15689:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15690:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15691:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15692:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15693:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
      TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15694:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
      lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
      : DWORD):BOOL');
15695:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
      lpcchBuffer:DWORD):BOOL');
15696:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
      lpcchBuffer:DWORD):BOOL;
15697:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15698:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15699:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15700:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15701:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
      cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15702:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15703:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15704:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15705:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15706: end;
15707:
15708: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15709: begin
15710:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
      AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
      BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15711:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
      DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15712:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15713:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
      TWinControl;
15714:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
      AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15715:  Function CreateNotifyThread(const
      FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15716: end;
15717:
15718: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15719: begin
15720:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15721:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15722:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15723:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15724:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
```

```
15725:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15726:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15727:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15728:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15729:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15730:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15731:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15732:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15733:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15734:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15735:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15736:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15737:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15738:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15739:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15740:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15741:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15742:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15743:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15744:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15745:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15746:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15747:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15748:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15749:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15750:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15751:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15752:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15753:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15754:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15755:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15756:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15757:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15758:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15759:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15760:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15761:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15762:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15763:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15764:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15765:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
      List:TStrings):Bool
15766:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15767:   FindClass('TOBJECT'),'EJclFileSummaryError');
15768:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15769:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15770:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15771:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15772:   SIRegister_TJclFilePropertySet(CL);
15773:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15774:   SIRegister_TJclFileSummary(CL);
15775:   SIRegister_TJclFileSummaryInformation(CL);
15776:   SIRegister_TJclDocSummaryInformation(CL);
15777:   SIRegister_TJclMediaFileSummaryInformation(CL);
15778:   SIRegister_TJclMSISummaryInformation(CL);
15779:   SIRegister_TJclShellSummaryInformation(CL);
15780:   SIRegister_TJclStorageSummaryInformation(CL);
15781:   SIRegister_TJclImageSummaryInformation(CL);
15782:   SIRegister_TJclDisplacedSummaryInformation(CL);
15783:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15784:   SIRegister_TJclMiscSummaryInformation(CL);
15785:   SIRegister_TJclWebViewSummaryInformation(CL);
15786:   SIRegister_TJclMusicSummaryInformation(CL);
15787:   SIRegister_TJclDRMSummaryInformation(CL);
15788:   SIRegister_TJclVideoSummaryInformation(CL);
15789:   SIRegister_TJclAudioSummaryInformation(CL);
15790:   SIRegister_TJclControlPanelSummaryInformation(CL);
15791:   SIRegister_TJclVolumeSummaryInformation(CL);
15792:   SIRegister_TJclShareSummaryInformation(CL);
15793:   SIRegister_TJclLinkSummaryInformation(CL);
15794:   SIRegister_TJclQuerySummaryInformation(CL);
15795:   SIRegister_TJclImageInformation(CL);
15796:   SIRegister_TJclJpegSummaryInformation(CL);
15797: end;
15798:
15799: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15800: begin
15801:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15802:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15803:   T8087Infinity','( icProjective, icAffine )');
15804:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15805:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15806:  Function Get8087ControlWord : Word');
15807:  Function Get8087Infinity : T8087Infinity');
15808:  Function Get8087Precision : T8087Precision');
15809:  Function Get8087Rounding : T8087Rounding');
15810:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15811:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15812:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
```

```
15813:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15814:  Function Set8087ControlWord( const Control : Word) : Word');
15815:  Function ClearPending8087Exceptions : T8087Exceptions');
15816:  Function GetPending8087Exceptions : T8087Exceptions');
15817:  Function GetMasked8087Exceptions : T8087Exceptions');
15818:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15819:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15820:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15821: end;
15822:
15823: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15824: begin
15825:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15826:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15827:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
       Accept:Bool;Sorted:Bool;
15828:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15829:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15830:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15831:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15832:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15833: end;
15834:
15835: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15836: begin
15837:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15838:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15839:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15840:   type ULONG', 'Cardinal');
15841:      LPCWSTR', 'PChar');
15842:   CL.AddTypeS('LPWSTR', 'PChar');
15843:   LPSTR', 'PChar');
15844:   TBindVerb', 'ULONG');
15845:   TBindInfoF', 'ULONG');
15846:   TBindF', 'ULONG');
15847:   TBSCF', 'ULONG');
15848:   TBindStatus', 'ULONG');
15849:   TCIPStatus', 'ULONG');
15850:   TBindString', 'ULONG');
15851:   TPiFlags', 'ULONG');
15852:   TOIBdgFlags', 'ULONG');
15853:   TParseAction', 'ULONG');
15854:   TPSUAction', 'ULONG');
15855:   TQueryOption', 'ULONG');
15856:   TPUAF', 'ULONG');
15857:   TSZMFlags', 'ULONG');
15858:   TUrlZone', 'ULONG');
15859:   TUrlTemplate', 'ULONG');
15860:   TZAFlags', 'ULONG');
15861:   TUrlZoneReg', 'ULONG');
15862:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15863:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15864:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15865:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15866:  const 'CF_NULL','LongInt').SetInt( 0);
15867:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15868:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15869:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15870:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15871:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15872:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15873:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15874:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15875:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15876:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15877:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15878:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15879:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15880:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15881:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15882:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15883:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15884:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15885:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15886:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15887:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15888:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15889:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15890:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15891:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15892:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15893:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15894:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15895:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15896:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15897:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15898:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15899:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15900:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
```

```
15901:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15902:   SIRegister_IPersistMoniker(CL);
15903:   SIRegister_IBindProtocol(CL);
15904:   SIRegister_IBinding(CL);
15905:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15906:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15907:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15908:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15909:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15910:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15911:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15912:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15913:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
15914:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15915:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15916:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15917:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
15918:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
15919:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
15920:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
15921:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
15922:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
15923:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
15924:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
15925:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
15926:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
15927:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
15928:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
15929:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
15930:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
15931:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
15932:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
15933:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
15934:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
15935:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
15936:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
15937:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
15938:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
15939:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
15940:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
15941:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
15942:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
15943:  const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
15944:  const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
15945:  const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
15946:  const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
15947:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
15948:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
15949:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
15950:  const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
15951:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
15952:  const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
15953:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
15954:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
15955:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
15956:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
15957:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
15958:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
15959:  const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
15960:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
15961:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
15962:  const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
15963:  const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
15964:  const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
15965:  const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
15966:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
15967:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
15968:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
15969:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
15970:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
15971:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
15972:  const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
15973:  const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
15974:  const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
15975:  const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
15976:  const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
15977:  const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
15978:  const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
15979:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
15980:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
15981:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
15982:  const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
15983:  const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
15984:  const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
15985:  const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
15986:  const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
15987:  const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
15988:  const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
15989:  // PBindInfo', '^TBindInfo // will not work');
```

```
15990:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
15991:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
15992:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
15993:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
15994:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
15995:   TBindInfo', '_tagBINDINFO');
15996:   BINDINFO', '_tagBINDINFO');}
15997:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
15998:    +'criptor : DWORD; bInheritHandle : BOOL; end');
15999:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16000:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16001:   //PRemBindInfo', '^TRemBindInfo // will not work');
16002:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16003:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16004:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16005:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16006:    +'n; dwReserved : DWORD; end');
16007:   TRemBindInfo', '_tagRemBINDINFO');
16008:   RemBINDINFO', '_tagRemBINDINFO');}
16009:   //PRemFormatEtc', '^TRemFormatEtc // will not work');
16010:   tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16011:   TRemFormatEtc', 'tagRemFORMATETC');
16012:   RemFORMATETC', 'tagRemFORMATETC');
16013:   SIRegister_IBindStatusCallback(CL);
16014:   SIRegister_IAuthenticate(CL);
16015:   SIRegister_IHttpNegotiate(CL);
16016:   SIRegister_IWindowForBindingUI(CL);
16017:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16018:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16019:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16020:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16021:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16022:  const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16023:  const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT',LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16024:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16025:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16026:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16027:   SIRegister_ICodeInstall(CL);
16028:   SIRegister_IWinInetInfo(CL);
16029:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16030:   SIRegister_IHttpSecurity(CL);
16031:   SIRegister_IWinInetHttpInfo(CL);
16032:   SIRegister_IBindHost(CL);
16033:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16034:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16035:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16036:  Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback) : HResult');
16037:  Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback) : HResult');
16038:  Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
      : IBindStatusCallback) : HResult');
16039:  Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
     IBindStatusCallback) : HResult');
16040:  Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
     p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16041:  Function HlinkGoBack( unk : IUnknown) : HResult');
16042:  Function HlinkGoForward( unk : IUnknown) : HResult');
16043:  Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult');
16044: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult');
16045:   SIRegister_IInternet(CL);
16046:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16047:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16048:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16049:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16050:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16051:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16052:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16053:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16054:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16055:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16056:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16057:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16058:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16059:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16060:   //POLEStrArray', '^TOLESTRArray // will not work');
16061:   SIRegister_IInternetBindInfo(CL);
16062:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16063:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16064:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16065:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16066:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16067:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16068:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16069:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16070:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16071:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16072:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16073:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16074:   //PProtocolData', '^TProtocolData // will not work');
16075:   _tagPROTOCOLDATA','record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
```

```
16076:   TProtocolData', '_tagPROTOCOLDATA');
16077:   PROTOCOLDATA', '_tagPROTOCOLDATA');
16078:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink, 'IInternetProtocolSink');
16079:   SIRegister_IInternetProtocolRoot(CL);
16080:   SIRegister_IInternetProtocol(CL);
16081:   SIRegister_IInternetProtocolSink(CL);
16082:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16083:   SIRegister_IInternetSession(CL);
16084:   SIRegister_IInternetThreadSwitch(CL);
16085:   SIRegister_IInternetPriority(CL);
16086:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16087:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16088:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16089:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16090:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16091:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16092:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16093:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16094:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16095:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16096:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16097:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16098:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16099:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16100:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16101:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16102:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16103:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16104:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16105:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16106:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16107:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16108:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16109:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16110:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16111:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16112:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16113:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16114:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16115:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16116:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16117:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16118:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16119:   SIRegister_IInternetProtocolInfo(CL);
16120:   IOInet', 'IInternet');
16121:   IOInetBindInfo', 'IInternetBindInfo');
16122:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16123:   IOInetProtocol', 'IInternetProtocol');
16124:   IOInetProtocolSink', 'IInternetProtocolSink');
16125:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16126:   IOInetSession', 'IInternetSession');
16127:   IOInetPriority', 'IInternetPriority');
16128:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16129:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
      LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16130:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
      LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16131:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16132:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16133:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
      pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16134:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
      IInternetSes;dwReserved:DWORD):HResult;
16135:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
      pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16136:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
      LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16137:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
      LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16138:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16139:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16140:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
      TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16141:  Function OInetGetSession(dwSessionMode:DWORD; var
      pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16142:    //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16143:   //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16144: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16145: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16146:   //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16147:   //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16148:   //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16149:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16150:   SIRegister_IInternetSecurityMgrSite(CL);
16151:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16152:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16153:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16154:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16155:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
```

```
16156:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16157:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16158:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16159:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16160:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16161:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16162:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16163:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16164:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16165:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16166:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16167:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16168:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16169:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16170:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16171:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16172:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16173:  const 'PUAF_NOUIFLOCKED','LongWord').SetUInt( $00100000);
16174:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16175:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16176:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16177:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16178:   SIRegister_IInternetSecurityManager(CL);
16179:   SIRegister_IInternetHostSecurityManager(CL);
16180:   SIRegister_IInternetSecurityManagerEx(CL);
16181:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16182:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16183:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16184:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16185:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16186:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16187:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16188:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16189:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16190:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16191:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16192:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16193:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16194:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16195:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16196:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16197:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16198:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16199:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16200:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16201:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16202:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16203:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16204:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16205:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16206:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16207:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16208:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16209:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16210:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16211:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16212:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16213:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16214:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16215:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16216:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16217:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16218:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16219:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16220:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16221:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16222:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16223:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16224:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16225:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16226:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16227:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16228:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16229:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16230:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16231:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16232:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16233:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16234:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16235:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16236:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16237:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16238:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16239:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16240:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16241:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16242:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16243:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16244:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
```

```
16245:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16246:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16247:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16248:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16249:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16250:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16251:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16252:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16253:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16254:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16255:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16256:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16257:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16258:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16259:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16260:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16261:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16262:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16263:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16264:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16265:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16266:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16267:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16268:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16269:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16270:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16271:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16272:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16273:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16274:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16275:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16276:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16277:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16278:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16279:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16280:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16281:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16282:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16283:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16284:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16285:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16286:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16287:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16288:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16289:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16290:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16291:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16292:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16293:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16294:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16295:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16296:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16297:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16298:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16299:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16300:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16301:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16302:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16303:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16304:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16305:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16306:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16307:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16308:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16309:   //PZoneAttributes', '^TZoneAttributes // will not work');
16310:  _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');
16311:   { _ZONEATTRIBUTES = packed record
16312:     cbSize: ULONG;
16313:     szDisplayName: array [0..260 - 1] of WideChar;
16314:     szDescription: array [0..200 - 1] of WideChar;
16315:     szIconPath: array [0..260 - 1] of WideChar;
16316:     dwTemplateMinLevel: DWORD;
16317:     dwTemplateRecommended: DWORD;
16318:     dwTemplateCurrentLevel: DWORD;
16319:     dwFlags: DWORD;
16320:   end; }
16321:   TZoneAttributes', '_ZONEATTRIBUTES');
16322:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16323:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16324:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16325:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16326:   SIRegister_IInternetZoneManager(CL);
16327:   SIRegister_IInternetZoneManagerEx(CL);
16328:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16329:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16330:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16331:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
```

```
16332:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16333:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16334:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16335:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16336:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16337:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16338:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16339:   TCodeBaseHold', '_tagCODEBASEHOLD');
16340:   CODEBASEHOLD', '_tagCODEBASEHOLD');
16341:   //PSoftDistInfo', '^TSoftDistInfo // will not work');
16342:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16343:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16344:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
16345:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16346:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16347:   TSoftDistInfo', '_tagSOFTDISTINFO');
16348:   SOFTDISTINFO', '_tagSOFTDISTINFO');
16349:   SIRegister_ISoftDistExt(CL);
16350:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16351:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
       dwAdvertisedVersionLS : DWORD) : HResult');
16352:   SIRegister_IDataFilter(CL);
16353:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16354:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
16355:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16356:    +'terFlags : DWORD; end');
16357:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16358:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16359:   //PDataInfo', '^TDataInfo // will not work');
16360:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16361:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16362:   TDataInfo', '_tagDATAINFO');
16363:   DATAINFO', '_tagDATAINFO');
16364:   SIRegister_IEncodingFilterFactory(CL);
16365:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16366:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16367:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16368:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16369:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16370:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16371:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16372:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16373:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16374: end;
16375:
16376: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16377: begin
16378:  Procedure reformatMemo( const m : TCustomMemo)');
16379:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16380:  Procedure MoveToTop( memo : TMemo)');
16381:  Procedure ScrollToTop( memo : TMemo)');
16382:  Function LineNumberClicked( memo : TMemo) : integer');
16383:  Function MemoClickedLine( memo : TMemo) : integer');
16384:  Function ClickedMemoLine( memo : TMemo) : integer');
16385:  Function MemoLineClicked( memo : TMemo) : integer');
16386:  Function LinePositionClicked( Memo : TMemo) : integer');
16387:  Function ClickedMemoPosition( memo : TMemo) : integer');
16388:  Function MemoPositionClicked( memo : TMemo) : integer');
16389:  Procedure AdjustGridSize( grid : TDrawGrid)');
16390:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16391:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16392:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16393:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16394:  Procedure sortstrDown( var s : string)');
16395:  Procedure sortstrUp( var s : string)');
16396:  Procedure rotatestrleft( var s : string)');
16397:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16398:  Function deblank( s : string) : string');
16399:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16400:  Procedure FreeAndClearListBox( C : TListBox);');
16401:  Procedure FreeAndClearMemo( C : TMemo);');
16402:  Procedure FreeAndClearStringList( C : TStringList);');
16403:  Function dffgetfilesize( f : TSearchrec) : int64');
16404: end;
16405:
16406: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16407: begin
16408:   CL.AddTypeS('intset', 'set of byte');
16409:   TPoint64', 'record x : int64; y : int64; end');
16410:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16411:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16412:  Function GeneratePentagon( n : integer) : integer');
16413:  Function IsPentagon( p : integer) : boolean');
16414:  Function isSquare( const N : int64) : boolean');
16415:  Function isCube( const N : int64) : boolean');
16416:  Function isPalindrome( const n : int64) : boolean;');
16417:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16418:  Function GetEulerPhi( n : int64) : int64');
16419:  Function dffIntPower( a, b : int64) : int64;');
```

```
16420:  Function IntPower1( a : extended; b : int64) : extended;');
16421:  Function gcd2( a, b : int64) : int64');
16422:  Function GCDMany( A : array of integer) : integer');
16423:  Function LCMMany( A : array of integer) : integer');
16424:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16425:  Function dffFactorial( n : int64) : int64');
16426:  Function digitcount( n : int64) : integer');
16427:  Function nextpermute( var a : array of integer) : boolean');
16428:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16429:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16430:  Function InttoBinaryStr( nn : integer) : string');
16431:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16432:  Function AngleToStr( angle : extended) : string');
16433:  Function deg2rad( deg : extended) : extended');
16434:  Function rad2deg( rad : extended) : extended');
16435:  Function GetLongToMercProjection( const long : extended) : extended');
16436:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16437:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16438:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16439:   SIRegister_TPrimes(CL);
16440:   //RIRegister_TPrimes(CL);
16441:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16442:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16443:  Function Random64( const N : Int64) : Int64;');
16444:  Procedure Randomize64');
16445:  Function Random641 : extended;');
16446:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)');
16447: end;
16448:
16449: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16450: begin
16451:   TrealPoint', 'record x : extended; y : extended; end');
16452:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16453:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16454:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16455:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16456:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16457:  Function realpoint( x, y : extended) : TRealPoint');
16458:  Function dist( const p1, p2 : TrealPoint) : extended');
16459:  Function intdist( const p1, p2 : TPoint) : integer');
16460:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16461:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16462:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16463:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16464:  Function GetTheta( const L : TLine) : extended;');
16465:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16466:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16467:  Procedure Extendline( var L : TLine; dist : integer);');
16468:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16469:  Function Linesintersect( line1, line2 : TLine) : boolean');
16470:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16471:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16472:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16473:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16474:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16475:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16476:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16477:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16478:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16479:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16480:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16481:  Function DegtoRad( d : extended) : extended');
16482:  Function RadtoDeg( r : extended) : extended');
16483:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16484:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16485:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16486:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16487:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16488:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16489:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16490:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16491:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16492: end;
16493:
16494:
16495: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16496: begin
16497:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16498:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16499:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16500:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16501:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16502:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16503:    +'arth : extended; Phase : extended; end');
```

```
16504:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16505:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16506:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16507:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16508:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16509:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16510:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16511:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16512:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16513:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
       ApparentRaDecl:TRPoint; end');
16514:   SIRegister_TAstronomy(CL);
16515:  Function AngleToStr( angle : extended) : string');
16516:  Function StrToAngle( s : string; var angle : extended) : boolean');
16517:  Function HoursToStr24( t : extended) : string');
16518:  Function RPoint( x, y : extended) : TRPoint');
16519:  Function getStimename( t : TDTType) : string');
16520: end;
16521:
16522: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16523: begin
16524:   TCardValue', 'Integer');
16525:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16526:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16527:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16528:   SIRegister_TCard(CL);
16529:   SIRegister_TDeck(CL);
16530: end;
16531:
16532: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16533: begin
16534:   tMethodCall', 'Procedure');
16535:   tVerboseCall', 'Procedure ( s : string)');
16536:  // PTEdge', '^TEdge // will not work');
16537:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16538:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16539:   SIRegister_TNode(CL);
16540:   SIRegister_TGraphList(CL);
16541: end;
16542:
16543: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16544: begin
16545:   ParserFloat', 'extended');
16546:   //PParserFloat', '^ParserFloat // will not work');
16547:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16548:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16549:   //POperation', '^TOperation // will not work');
16550:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16551:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16552:    +'; Token : TDFFToken; end');
16553:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16554:   (CL.FindClass('TOBJECT'),'EMathParserError');
16555:   CL.FindClass('TOBJECT'),'ESyntaxError');
16556:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16557:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16558:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16559:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16560:   (CL.FindClass('TOBJECT'),'EBadName');
16561:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16562:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16563:   SIRegister_TCustomParser(CL);
16564:   SIRegister_TExParser(CL);
16565: end;
16566:
16567:   function isService: boolean;
16568:   begin
16569:     result:= NOT(Application is TApplication);
16570:     {result:= Application is TServiceApplication;}
16571:   end;
16572:   function isApplication: boolean;
16573:   begin
16574:     result:= Application is TApplication;
16575:   end;
16576:   //SM_REMOTESESSION = $1000
16577:   function isTerminalSession: boolean;
16578:   begin
16579:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16580:   end;
16581:
16582: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16583: begin
16584:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16585:    +'String; margin_bottom : String; margin_left : String; margin_right : Strin'
16586:    +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16587:  Function cyURLEncode( const S : string) : string');
16588:  Function MakeResourceURL(const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16589:  Function MakeResourceURL1(const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;;
16590:  Function cyColorToHtml( aColor : TColor) : String');
```

```
16591:  Function HtmlToColor( aHtmlColor : String) : TColor');
16592:  //Function GetStreamEncoding( aStream : TStream) : TEncoding');
16593:  // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean');
16594:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
16595:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16596:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16597:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16598:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16599:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16600:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16601:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16602:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16603:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16604: end;
16605:
16606:
16607: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16608: begin
16609:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16610:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16611:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16612:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16613:    +'inationsRepeat, CombinationsRepeatDown )');
16614:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16615:   SIRegister_TComboSet(CL);
16616: end;
16617:
16618:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16619: begin
16620:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
16621:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16622:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16623:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16624:    +'mmHandle : THandle; aString : String; userParam : Integer)');
16625:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16626:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16627:   SIRegister_TcyBaseComm(CL);
16628:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16629:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16630:  Function ValidateFileMappingName( aName : String) : String');
16631:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16632: end;
16633:
16634: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16635: begin
16636:   CL.AddTypeS('DERString', 'String');
16637:   CL.AddTypeS('DERChar', 'Char');
16638:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16639:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16640:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16641:   CL.AddTypeS('DERNString', 'String');
16642:  const DERDecimalSeparator','String').SetString( '.');
16643:  const DERDefaultChars','String')('+°@/%-
      _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16644:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16645:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16646:  Function isValidWebMailChar( aChar : Char) : Boolean');
16647:  Function isValidwebSite( aStr : String) : Boolean');
16648:  Function isValidWebMail( aStr : String) : Boolean');
16649:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16650:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16651:  Function IsDERChar( aChar : Char) : Boolean');
16652:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16653:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16654:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16655:  Function IsDERSymbols( aDERString : String) : Boolean');
16656:  Function StringToDERCharSet( aStr : String) : DERString');
16657:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16658:  Function IsDERNChar( aChar : Char) : Boolean');
16659:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16660:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
16661:  Function DERExtractWebMail( aDERStr : DERString) : String');
16662:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16663:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16664:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
      RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16665:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
      aElementsType : TElementsType) : String;');
16666:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
      Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16667: end;
16668:
16669: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16670: begin
16671:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16672:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16673:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
```

```
16674:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16675:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16676:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16677:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16678:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16679:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
       RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16680:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
       PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
       PercentRange2Blue:Double;SingleDestinationColor : Boolean;RefreshBmp:Bool;');
16681:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
       SingleDestinationColor, RefreshBmp : Boolean)');
16682:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16683:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16684:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16685:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16686:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16687:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16688:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16689:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16690: end;
16691:
16692:
16693: {A simple Oscilloscope using  TWaveIn class.
16694:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
16695: uses
16696:   Forms,
16697:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
16698:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
16699:   uColorFunctions in 'uColorFunctions.pas',
16700:   AMixer in 'AMixer.pas',
16701:   uSettings in 'uSettings.pas',
16702:   UWavein4 in 'UWavein4.pas',
16703:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
16704:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
16705:
16706:
16707: Functions_max hex in the box maXbox
16708: functionslist.txt
16709: FunctionsList1 3.9.9.86/88/91/92/94/95/96
16710:
16711: ********************************************************************************
16712: Procedure
16713: PROCEDURE SIZE 7507 7401 6792 6310 5971 4438 3797 3600
16714: Procedure ***************************Now the Procedure list******************
16715: Procedure ( ACol, ARow : Integer; Items : TStrings)
16716: Procedure ( Agg : TAggregate)
16717: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
16718: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
16719: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
16720: Procedure ( ASender : TObject; const ABytes : Integer)
16721: Procedure ( ASender : TObject; VStream : TStream)
16722: Procedure ( AThread : TIdThread)
16723: Procedure ( AWebModule : TComponent)
16724: Procedure ( Column : TColumn)
16725: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
16726: Procedure ( const iStart : integer; const sText : string)
16727: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
16728: Procedure ( Database : TDatabase; LoginParams : TStrings)
16729: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
16730: Procedure ( DATASET : TDATASET)
16731: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
16732: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
16733: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
16734: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
16735: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
16736: Procedure ( Done : Integer)
16737: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
16738: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
16739: Procedure
       (HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
16740: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
16741: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
16742: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
16743: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
16744: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
16745: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
16746: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
16747: Procedure ( SENDER : TFIELD; const TEXT : String)
16748: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
16749: Procedure ( Sender : TIdTelnet; const Buffer : String)
16750: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
16751: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
16752: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
16753: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
16754: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
16755: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
16756: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
```

```
16757: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
16758: Procedure ( Sender : TObject; Button : TMPBtnType)
16759: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
16760: Procedure ( Sender : TObject; Button : TUDBtnType)
16761: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
16762: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
16763: Procedure ( Sender : TObject; Column : TListColumn)
16764: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
16765: Procedure ( Sender : TObject; Connecting : Boolean)
16766: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
16767: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
16768: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
16769: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
16770: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
16771: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
16772: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
16773: Procedure ( Sender : TObject; Index : LongInt)
16774: Procedure ( Sender : TObject; Item : TListItem)
16775: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
16776: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
16777: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
16778: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
16779: Procedure ( Sender : TObject; Item : TListItem; var S : string)
16780: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
16781: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
16782: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
16783: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
16784: Procedure ( Sender : TObject; Node : TTreeNode)
16785: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
16786: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
16787: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
16788: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
16789: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
16790: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
16791: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
16792: Procedure ( Sender : TObject; Rect : TRect)
16793: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
16794: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
16795: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
16796: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
16797: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
16798: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
16799: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
16800: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
16801: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
16802: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
16803: Procedure ( Sender : TObject; Thread : TServerClientThread)
16804: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
16805: Procedure ( Sender : TObject; Username, Password : string)
16806: Procedure ( Sender : TObject; var AllowChange : Boolean)
16807: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
16808: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
16809: Procedure ( Sender : TObject; var Continue : Boolean)
16810: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
16811: Procedure ( Sender : TObject; var Username : string)
16812: Procedure ( Sender : TObject; Wnd : HWND)
16813: Procedure ( Sender : TToolbar; Button : TToolButton)
16814: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
16815: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
16816: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
16817: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
16818: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
16819: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
16820: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
16821: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
16822: procedure (Sender: TObject)
16823: procedure (Sender: TObject; var Done: Boolean)
16824: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
16825: procedure _T(Name: tbtString; v: Variant);
16826: Procedure AbandonSignalHandler( RtlSigNum : Integer)
16827: Procedure Abort
16828: Procedure About1Click( Sender : TObject)
16829: Procedure Accept( Socket : TSocket)
16830: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
16831: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
16832: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
16833: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
16834: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
16835: Procedure Add( Addend1, Addend2 : TMyBigInt)
16836: Procedure ADD( const AKEY, AVALUE : VARIANT)
16837: Procedure Add( const Key : string; Value : Integer)
16838: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
16839: Procedure ADD( FIELD : TFIELD)
16840: Procedure ADD( ITEM : TMENUITEM)
16841: Procedure ADD( POPUP : TPOPUPMENU)
16842: Procedure AddCharacters( xCharacters : TCharSet)
16843: Procedure AddDriver( const Name : string; List : TStrings)
16844: Procedure AddImages( Value : TCustomImageList)
```

```
16845: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
16846: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
16847: Procedure AddLoader( Loader : TBitmapLoader)
16848: Procedure ADDPARAM( VALUE : TPARAM)
16849: Procedure AddPassword( const Password : string)
16850: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
16851: Procedure AddState( oState : TniRegularExpressionState)
16852: Procedure AddStrings( Strings : TStrings);
16853: procedure AddStrings(Strings: TStrings);
16854: Procedure AddStrings1( Strings : TWideStrings);
16855: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
16856: Procedure AddToRecentDocs( const Filename : string)
16857: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
16858: Procedure AllFunctionsList1Click( Sender : TObject)
16859: procedure AllObjectsList1Click(Sender: TObject);
16860: Procedure Allocate( AAllocateBytes : Integer)
16861: procedure AllResourceList1Click(Sender: TObject);
16862: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
16863: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
16864: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
16865: Procedure AnsiFree( var s : AnsiString)
16866: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
16867: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
16868: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
16869: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
16870: Procedure AntiFreeze;
16871: Procedure APPEND
16872: Procedure Append( const S : WideString)
16873: procedure Append(S: string);
16874: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
16875: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
16876: Procedure AppendChunk( Val : OleVariant)
16877: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
16878: Procedure AppendStr( var Dest : string; S : string)
16879: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
16880: Procedure ApplyRange
16881: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
16882: Procedure Arrange( Code : TListArrangement)
16883: procedure Assert(expr : Boolean; const msg: string);
16884: procedure Assert2(expr : Boolean; const msg: string);
16885: Procedure Assign( AList : TCustomBucketList)
16886: Procedure Assign( Other : TObject)
16887: Procedure Assign( Source : TDragObject)
16888: Procedure Assign( Source : TPersistent)
16889: Procedure Assign(Source: TPersistent)
16890: procedure Assign2(mystring, mypath: string);
16891: Procedure AssignCurValues( Source : TDataSet);
16892: Procedure AssignCurValues1( const CurValues : Variant);
16893: Procedure ASSIGNFIELD( FIELD : TFIELD)
16894: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
16895: Procedure AssignFile(var F: Text; FileName: string)
16896: procedure AssignFile(var F: TextFile; FileName: string)
16897: procedure AssignFileRead(var mystring, myfilename: string);
16898: procedure AssignFileWrite(mystring, myfilename: string);
16899: Procedure AssignTo( Other : TObject)
16900: Procedure AssignValues( Value : TParameters)
16901: Procedure ASSIGNVALUES( VALUE : TPARAMS)
16902: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
16903: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
16904: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
16905: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
16906: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
16907: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
16908: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
16909: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
16910: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
16911: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
16912: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
16913: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
16914: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
16915: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
16916: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
16917: procedure Beep
16918: Procedure BeepOk
16919: Procedure BeepQuestion
16920: Procedure BeepHand
16921: Procedure BeepExclamation
16922: Procedure BeepAsterisk
16923: Procedure BeepInformation
16924: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
16925: Procedure BeginLayout
16926: Procedure BeginTimer( const Delay, Resolution : Cardinal)
16927: Procedure BeginUpdate
16928: procedure BeginUpdate;
16929: procedure BigScreen1Click(Sender: TObject);
16930: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
16931: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
16932: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
16933: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
```

```
16934: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
16935: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
16936: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
16937: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
16938: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
16939: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
16940: Procedure BreakPointMenuClick( Sender : TObject)
16941: procedure BRINGTOFRONT
16942: procedure BringToFront;
16943: Procedure btnBackClick( Sender : TObject)
16944: Procedure btnBrowseClick( Sender : TObject)
16945: Procedure BtnClick( Index : TNavigateBtn)
16946: Procedure btnLargeIconsClick( Sender : TObject)
16947: Procedure BuildAndSendRequest( AURI : TIdURI)
16948: Procedure BuildCache
16949: Procedure BurnMemory( var Buff, BuffLen : integer)
16950: Procedure BurnMemoryStream( Destructo : TMemoryStream)
16951: Procedure CalculateFirstSet
16952: Procedure Cancel
16953: procedure CancelDrag;
16954: Procedure CancelEdit
16955: procedure CANCELHINT
16956: Procedure CancelRange
16957: Procedure CancelUpdates
16958: Procedure CancelWriteBuffer
16959: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
16960: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
16961: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
16962: procedure CaptureScreenFormat(vname: string; vextension: string);
16963: procedure CaptureScreenPNG(vname: string);
16964: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
16965: procedure CASCADE
16966: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
16967: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
16968: Procedure cbPathClick( Sender : TObject)
16969: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
16970: Procedure cedebugAfterExecute( Sender : TPSScript)
16971: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
16972: Procedure cedebugCompile( Sender : TPSScript)
16973: Procedure cedebugExecute( Sender : TPSScript)
16974: Procedure cedebugIdle( Sender : TObject)
16975: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
16976: Procedure CenterHeight( const pc, pcParent : TControl)
16977: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
16978: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
16979: Procedure Change
16980: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
16981: Procedure Changed
16982: Procedure ChangeDir( const ADirName : string)
16983: Procedure ChangeDirUp
16984: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
16985: Procedure ChangeLevelBy( Value : TChangeRange)
16986: Procedure ChDir(const s: string)
16987: Procedure Check(Status: Integer)
16988: Procedure CheckCommonControl( CC : Integer)
16989: Procedure CHECKFIELDNAME( const FIELDNAME : String)
16990: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
16991: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
16992: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
16993: Procedure CheckToken( T : Char)
16994: procedure CheckToken(t:char)
16995: Procedure CheckTokenSymbol( const S : string)
16996: procedure CheckTokenSymbol(s:string)
16997: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
16998: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
16999: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
17000: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
17001: procedure CipherFile1Click(Sender: TObject);
17002: Procedure Clear;
17003: Procedure Clear1Click( Sender : TObject)
17004: Procedure ClearColor( Color : TColor)
17005: Procedure CLEARITEM( AITEM : TMENUITEM)
17006: Procedure ClearMapping
17007: Procedure ClearSelection( KeepPrimary : Boolean)
17008: Procedure ClearWriteBuffer
17009: Procedure Click
17010: Procedure Close
17011: Procedure Close1Click( Sender : TObject)
17012: Procedure CloseDatabase( Database : TDatabase)
17013: Procedure CloseDataSets
17014: Procedure CloseDialog
17015: Procedure CloseFile(var F: Text);
17016: Procedure Closure
17017: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17018: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17019: Procedure CodeCompletionList1Click( Sender : TObject)
17020: Procedure ColEnter
17021: Procedure Collapse
17022: Procedure Collapse( Recurse : Boolean)
```

```
17023: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
17024: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
17025: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
17026: Procedure Compile1Click( Sender : TObject)
17027: procedure ComponentCount1Click(Sender: TObject);
17028: Procedure Compress(azipfolder, azipfile: string)
17029: Procedure DeCompress(azipfolder, azipfile: string)
17030: Procedure XZip(azipfolder, azipfile: string)
17031: Procedure XUnZip(azipfolder, azipfile: string)
17032: Procedure Connect( const ATimeout : Integer)
17033: Procedure Connect( Socket : TSocket)
17034: procedure Console1Click(Sender: TObject);
17035: Procedure Continue
17036: Procedure ContinueCount( var Counter : TJclCounter)
17037: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
17038: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
17039: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
17040: Procedure ConvertImage(vsource, vdestination: string);
17041: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
17042: Procedure ConvertBitmap(vsource, vdestination: string);
17043: Procedure ConvertToGray(Cnv: TCanvas);
17044: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
17045: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
17046: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
17047: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
17048: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
17049: Procedure CopyFrom( mbCopy : TMyBigInt)
17050: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
17051: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
17052: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
17053: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
17054: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
17055: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
17056: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
17057: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
17058: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
17059: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17060: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
17061: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17062: Procedure CopyToClipboard
17063: Procedure CountParts
17064: Procedure CreateDataSet
17065: Procedure CreateEmptyFile( const FileName : string)
17066: Procedure CreateFileFromString( const FileName, Data : string)
17067: Procedure CreateFromDelta( Source : TPacketDataSet)
17068: procedure CREATEHANDLE
17069: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
17070: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
17071: Procedure CreateTable
17072: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
17073: procedure CSyntax1Click(Sender: TObject);
17074: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
17075: Procedure CURSORPOSCHANGED
17076: procedure CutFirstDirectory(var S: String)
17077: Procedure DataBaseError(const Message: string)
17078: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
17079: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
17080: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
17081: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
17082: Procedure DBIError(errorCode: Integer)
17083: Procedure DebugOutput( const AText : string)
17084: Procedure DebugRun1Click( Sender : TObject)
17085: procedure Dec;
17086: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
17087: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
17088: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
17089: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
17090: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
17091: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
17092: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
17093: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
17094: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
17095: Procedure Decompile1Click( Sender : TObject)
17096: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
17097: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
17098: Procedure DeferLayout
17099: Procedure defFileread
17100: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
17101: Procedure DelayMicroseconds( const MicroSeconds : Integer)
17102: Procedure Delete
17103: Procedure Delete( const AFilename : string)
17104: Procedure Delete( const Index : Integer)
17105: Procedure DELETE( INDEX : INTEGER)
17106: Procedure Delete( Index : LongInt)
17107: Procedure Delete( Node : TTreeNode)
17108: procedure Delete(var s: AnyString; ifrom, icount: Longint);
17109: Procedure DeleteAlias( const Name : string)
```

```
17110: Procedure DeleteDriver( const Name : string)
17111: Procedure DeleteIndex( const Name : string)
17112: Procedure DeleteKey( const Section, Ident : String)
17113: Procedure DeleteRecords
17114: Procedure DeleteRecords( AffectRecords : TAffectRecords)
17115: Procedure DeleteString( var pStr : String; const pDelStr : string)
17116: Procedure DeleteTable
17117: procedure DelphiSite1Click(Sender: TObject);
17118: Procedure Deselect
17119: Procedure Deselect( Node : TTreeNode)
17120: procedure DestroyComponents
17121: Procedure DestroyHandle
17122: Procedure Diff( var X : array of Double)
17123: procedure Diff(var X: array of Double);
17124: procedure DISABLEALIGN
17125: Procedure DisableConstraints
17126: Procedure Disconnect
17127: Procedure Disconnect( Socket : TSocket)
17128: Procedure Dispose
17129: procedure Dispose(P: PChar)
17130: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
17131: Procedure DoKey( Key : TDBCtrlGridKey)
17132: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17133: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17134: Procedure Dormant
17135: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
17136: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
17137: Procedure DoubleToComp( Value : Double; var Result : Comp)
17138: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
17139: procedure Draw(X, Y: Integer; Graphic: TGraphic);
17140: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
17141: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17142: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
17143: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17144: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
17145: procedure DrawFocusRect(const Rect: TRect);
17146: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
17147: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17148: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
17149: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
17150: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
17151: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
17152: Procedure DropConnections
17153: Procedure DropDown
17154: Procedure DumpDescription( oStrings : TStrings)
17155: Procedure DumpStateTable( oStrings : TStrings)
17156: Procedure EDIT
17157: Procedure EditButtonClick
17158: Procedure EditFont1Click( Sender : TObject)
17159: procedure Ellipse(X1, Y1, X2, Y2: Integer);
17160: Procedure Ellipse1( const Rect : TRect);
17161: Procedure EMMS
17162: Procedure Encode( ADest : TStream)
17163: procedure ENDDRAG(DROP:BOOLEAN)
17164: Procedure EndEdit( Cancel : Boolean)
17165: Procedure EndTimer
17166: Procedure EndUpdate
17167: Procedure EraseSection( const Section : string)
17168: Procedure Error( const Ident : string)
17169: procedure Error(Ident:Integer)
17170: Procedure ErrorFmt( const Ident : string; const Args : array of const)
17171: Procedure ErrorStr( const Message : string)
17172: procedure ErrorStr(Message:String)
17173: Procedure Exchange( Index1, Index2 : Integer)
17174: procedure Exchange(Index1, Index2: Integer);
17175: Procedure Exec( FileName, Parameters, Directory : string)
17176: Procedure ExecProc
17177: Procedure ExecSQL( UpdateKind : TUpdateKind)
17178: Procedure Execute
17179: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
17180: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
17181: Procedure ExecuteCommand(executeFile, paramstring: string)
17182: Procedure ExecuteShell(executeFile, paramstring: string)
17183: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
17184: Procedure ExitThread(ExitCode: Integer); stdcall;
17185: Procedure ExitProcess(ExitCode: Integer); stdcall;
17186: Procedure Expand( AUserName : String; AResults : TStrings)
17187: Procedure Expand( Recurse : Boolean)
17188: Procedure ExportClipboard1Click( Sender : TObject)
17189: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
17190: Procedure ExtractContentFields( Strings : TStrings)
17191: Procedure ExtractCookieFields( Strings : TStrings)
17192: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
17193: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
17194: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
```

```
17195: Procedure ExtractQueryFields( Strings : TStrings)
17196: Procedure FastDegToGrad
17197: Procedure FastDegToRad
17198: Procedure FastGradToDeg
17199: Procedure FastGradToRad
17200: Procedure FastRadToDeg
17201: Procedure FastRadToGrad
17202: Procedure FileClose( Handle : Integer)
17203: Procedure FileClose(handle: integer)
17204: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
17205: Procedure FileStructure( AStructure : TIdFTPDataStructure)
17206: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
17207: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
17208: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
17209: Procedure FillChar2(var X: PChar ; count: integer; value: char)
17210: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
17211: Procedure FillIPList
17212: procedure FillRect(const Rect: TRect);
17213: Procedure FillTStrings( AStrings : TStrings)
17214: Procedure FilterOnBookmarks( Bookmarks : array of const)
17215: procedure FinalizePackage(Module: HMODULE)
17216: procedure FindClose;
17217: procedure FindClose2(var F: TSearchRec)
17218: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
17219: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
17220: Procedure FindNearest( const KeyValues : array of const)
17221: Procedure FinishContext
17222: Procedure FIRST
17223: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
17224: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
17225: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
17226: Procedure FlushSchemaCache( const TableName : string)
17227: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
17228: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
17229: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
17230: Procedure FormActivate( Sender : TObject)
17231: procedure FormatLn(const format: String; const args: array of const);  //alias
17232: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
17233: Procedure FormCreate( Sender : TObject)
17234: Procedure FormDestroy( Sender : TObject)
17235: Procedure FormKeyPress( Sender : TObject; var Key : Char)
17236: procedure FormOutput1Click(Sender: TObject);
17237: Procedure FormToHtml( Form : TForm; Path : string)
17238: procedure FrameRect(const Rect: TRect);
17239: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
17240: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
17241: Procedure Free( Buffer : TRecordBuffer)
17242: Procedure Free( Buffer : TValueBuffer)
17243: Procedure Free;
17244: Procedure FreeAndNil(var Obj:TObject)
17245: Procedure FreeImage
17246: procedure FreeMem(P: PChar; Size: Integer)
17247: Procedure FreeTreeData( Tree : TUpdateTree)
17248: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
17249: Procedure FullCollapse
17250: Procedure FullExpand
17251: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
17252: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
17253: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
17254: Procedure Get1( AURL : string; const AResponseContent : TStream);
17255: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
17256: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
17257: Procedure GetAliasNames( List : TStrings)
17258: Procedure GetAliasParams( const AliasName : string; List : TStrings)
17259: Procedure GetApplicationsRunning( Strings : TStrings)
17260: Procedure getBox(aURL, extension: string);
17261: Procedure GetCommandTypes( List : TWideStrings)
17262: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
17263: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
17264: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
17265: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
17266: Procedure GetDatabaseNames( List : TStrings)
17267: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
17268: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
17269: Procedure GetDir(d: byte; var s: string)
17270: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
17271: Procedure GetDriverNames( List : TStrings)
17272: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
17273: Procedure GetDriverParams( const DriverName : string; List : TStrings)
17274: Procedure GetEMails1Click( Sender : TObject)
17275: Procedure getEnvironmentInfo;
17276: Function getEnvironmentString: string;
17277: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
17278: Procedure GetFieldNames( const TableName : string; List : TStrings)
17279: Procedure GetFieldNames( const TableName : string; List : TStrings);
17280: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
17281: Procedure GETFIELDNAMES( LIST : TSTRINGS)
17282: Procedure GetFieldNames1( const TableName : string; List : TStrings);
17283: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
```

```
17284: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
17285: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
17286: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
17287: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
17288: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
17289: Procedure GetFormatSettings
17290: Procedure GetFromDIB( var DIB : TBitmapInfo)
17291: Procedure GetFromHDIB( HDIB : HBitmap)
17292: Procedure GetIcon( Index : Integer; Image : TIcon);
17293: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
17294: Procedure GetIndexInfo( IndexName : string)
17295: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
17296: Procedure GetIndexNames( List : TStrings)
17297: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
17298: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
17299: Procedure GetIndexNames4( const TableName : string; List : TStrings);
17300: Procedure GetInternalResponse
17301: Procedure GETITEMNAMES( LIST : TSTRINGS)
17302: procedure GetMem(P: PChar; Size: Integer)
17303: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
17304: procedure GetPackageDescription(ModuleName: PChar): string)
17305: Procedure GetPackageNames( List : TStrings);
17306: Procedure GetPackageNames1( List : TWideStrings);
17307: Procedure GetParamList( List : TList; const ParamNames : WideString)
17308: Procedure GetProcedureNames( List : TStrings);
17309: Procedure GetProcedureNames( List : TWideStrings);
17310: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
17311: Procedure GetProcedureNames1( List : TStrings);
17312: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
17313: Procedure GetProcedureNames3( List : TWideStrings);
17314: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
17315: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
17316: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
17317: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
17318: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
17319: Procedure GetProviderNames( Names : TWideStrings);
17320: Procedure GetProviderNames( Proc : TGetStrProc)
17321: Procedure GetProviderNames1( Names : TStrings);
17322: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
17323: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
17324: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
17325: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
17326: Procedure GetSchemaNames( List : TStrings);
17327: Procedure GetSchemaNames1( List : TWideStrings);
17328: Procedure getScriptandRunAsk;
17329: Procedure getScriptandRun(ascript: string);
17330: Procedure getScript(ascript: string);   //alias
17331: Procedure getWebScript(ascript: string);   //alias
17332: Procedure GetSessionNames( List : TStrings)
17333: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
17334: Procedure GetStrings( List : TStrings)
17335: Procedure GetSystemTime; stdcall;
17336: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
17337: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
17338: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
17339: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
17340: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
17341: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
17342: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
17343: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
17344: Procedure GetVisibleWindows( List : Tstrings)
17345: Procedure GoBegin
17346: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
17347: Procedure GotoCurrent( Table : TTable)
17348: procedure GotoEnd1Click(Sender: TObject);
17349: Procedure GotoNearest
17350: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
17351: Procedure HandleException( E : Exception; var Handled : Boolean)
17352: procedure HANDLEMESSAGE
17353: procedure HandleNeeded;
17354: Procedure Head( AURL : string)
17355: Procedure Help( var AHelpContents : TStringList; ACommand : String)
17356: Procedure HexToBinary( Stream : TStream)
17357: procedure HexToBinary(Stream:TStream)
17358: Procedure HideDragImage
17359: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
17360: Procedure HideTraybar
17361: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
17362: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
17363: Procedure HookOSExceptions
17364: Procedure HookSignal( RtlSigNum : Integer)
17365: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
17366: Procedure HTMLSyntax1Click( Sender : TObject)
17367: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
17368: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
17369: Procedure ImportfromClipboard1Click( Sender : TObject)
17370: Procedure ImportfromClipboard2Click( Sender : TObject)
```

```
17371: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
17372: procedure Incb(var x: byte);
17373: Procedure Include1Click( Sender : TObject)
17374: Procedure IncludeOFF;  //preprocessing
17375: Procedure IncludeON;
17376: procedure Info1Click(Sender: TObject);
17377: Procedure InitAltRecBuffers( CheckModified : Boolean)
17378: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
17379: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
17380: Procedure InitData( ASource : TDataSet)
17381: Procedure InitDelta( ADelta : TPacketDataSet);
17382: Procedure InitDelta1( const ADelta : OleVariant);
17383: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
17384: Procedure Initialize
17385: procedure InitializePackage(Module: HMODULE)
17386: Procedure INITIATEACTION
17387: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
17388: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
17389: Procedure InitModule( AModule : TComponent)
17390: Procedure InitStdConvs
17391: Procedure InitTreeData( Tree : TUpdateTree)
17392: Procedure INSERT
17393: Procedure Insert( Index : Integer; AClass : TClass)
17394: Procedure Insert( Index : Integer; AComponent : TComponent)
17395: Procedure Insert( Index : Integer; AObject : TObject)
17396: Procedure Insert( Index : Integer; const S : WideString)
17397: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
17398: Procedure Insert(Index: Integer; const S: string);
17399: procedure Insert(Index: Integer; S: string);
17400: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
17401: procedure InsertComponent(AComponent:TComponent)
17402: procedure InsertControl(AControl: TControl);
17403: Procedure InsertIcon( Index : Integer; Image : TIcon)
17404: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
17405: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
17406: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
17407: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
17408: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
17409: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
17410: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
17411: Procedure InternalBeforeResolve( Tree : TUpdateTree)
17412: Procedure InvalidateModuleCache
17413: Procedure InvalidateTitles
17414: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
17415: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
17416: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
17417: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
17418: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
17419: procedure JavaSyntax1Click(Sender: TObject);
17420: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
17421: Procedure KillDataChannel
17422: Procedure Largefont1Click( Sender : TObject)
17423: Procedure LAST
17424: Procedure LaunchCpl( FileName : string)
17425: Procedure Launch( const AFile : string)
17426: Procedure LaunchFile( const AFile : string)
17427: Procedure LetFileList(FileList: TStringlist; apath: string);
17428: Procedure lineToNumber( xmemo : String; met : boolean)
17429: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
17430: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
17431: Procedure ListViewData( Sender : TObject; Item : TListItem)
17432: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
17433: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
17434: Procedure ListViewDblClick( Sender : TObject)
17435: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
17436: Procedure ListDLLExports(const FileName: string; List: TStrings);
17437: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
17438: procedure LoadBytecode1Click(Sender: TObject);
17439: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
17440: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
17441: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
17442: Procedure LoadFromFile( AFileName : string)
17443: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
17444: Procedure LoadFromFile( const FileName : string)
17445: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
17446: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
17447: Procedure LoadFromFile( const FileName : WideString)
17448: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
17449: Procedure LoadFromFile(const AFileName: string)
17450: procedure LoadFromFile(FileName: string);
17451: procedure LoadFromFile(FileName:String)
17452: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
17453: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
17454: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
17455: Procedure LoadFromStream( const Stream : TStream)
```

```
17456: Procedure LoadFromStream( S : TStream)
17457: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
17458: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
17459: Procedure LoadFromStream( Stream : TStream)
17460: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
17461: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
17462: procedure LoadFromStream(Stream: TStream);
17463: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
17464: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
17465: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
17466: Procedure LoadLastFile1Click( Sender : TObject)
17467: { LoadIcoToImage loads two icons from resource named NameRes,
17468:   into two image lists ALarge and ASmall}
17469: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
17470: Procedure LoadMemo
17471: Procedure LoadParamsFromIniFile( FFileName : WideString)
17472: Procedure Lock
17473: Procedure Login
17474: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
17475: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
17476: Procedure MakeCaseInsensitive
17477: Procedure MakeDeterministic( var bChanged : boolean)
17478: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
17479: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
17480: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
17481: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
17482: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
17483: Procedure SetRectComplexFormatStr( const S : string)
17484: Procedure SetPolarComplexFormatStr( const S : string)
17485: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
17486: Procedure MakeVisible
17487: Procedure MakeVisible( PartialOK : Boolean)
17488: Procedure Manual1Click( Sender : TObject)
17489: Procedure MarkReachable
17490: Procedure maXbox;  //shows the exe version data in a win box
17491: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
17492: Procedure Memo1Change( Sender : TObject)
17493: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
17494: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
17495: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
17496: procedure Memory1Click(Sender: TObject);
17497: Procedure MERGE( MENU : TMAINMENU)
17498: Procedure MergeChangeLog
17499: procedure MINIMIZE
17500: Procedure MinimizeMaxbox;
17501: Procedure MkDir(const s: string)
17502: Procedure mnuPrintFont1Click( Sender : TObject)
17503: procedure ModalStarted
17504: Procedure Modified
17505: Procedure ModifyAlias( Name : string; List : TStrings)
17506: Procedure ModifyDriver( Name : string; List : TStrings)
17507: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
17508: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
17509: Procedure Move( CurIndex, NewIndex : Integer)
17510: procedure Move(CurIndex, NewIndex: Integer);
17511: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
17512: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
17513: Procedure moveCube( o : TMyLabel)
17514: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
17515: procedure MoveTo(X, Y: Integer);
17516: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
17517: Procedure MovePoint(var x,y:Extended; const angle:Extended);
17518: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
17519: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
17520: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
17521: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
17522: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
17523: procedure New(P: PChar)
17524: procedure New1Click(Sender: TObject);
17525: procedure NewInstance1Click(Sender: TObject);
17526: Procedure NEXT
17527: Procedure NextMonth
17528: Procedure Noop
17529: Procedure NormalizePath( var APath : string)
17530: procedure ObjectBinaryToText(Input, Output: TStream)
17531: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17532: procedure ObjectResourceToText(Input, Output: TStream)
17533: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17534: procedure ObjectTextToBinary(Input, Output: TStream)
17535: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17536: procedure ObjectTextToResource(Input, Output: TStream)
17537: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17538: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
17539: Procedure Open( const UserID : WideString; const Password : WideString);
17540: Procedure Open;
17541: Procedure open1Click( Sender : TObject)
17542: Procedure OpenCdDrive
17543: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
```

```
17544: Procedure OpenCurrent
17545: Procedure OpenFile(vfilenamepath: string)
17546: Procedure OpenDirectory1Click( Sender : TObject)
17547: Procedure OpenIndexFile( const IndexName : string)
17548: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
17549: Procedure OpenWriteBuffer( const AThreshhold : Integer)
17550: Procedure OptimizeMem
17551: Procedure Options1( AURL : string);
17552: Procedure OutputDebugString(lpOutputString : PChar)
17553: Procedure PackBuffer
17554: Procedure Paint
17555: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
17556: Procedure PaintToTBitmap( Target : TBitmap)
17557: Procedure PaletteChanged
17558: Procedure ParentBiDiModeChanged
17559: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
17560: Procedure PasteFromClipboard;
17561: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
17562: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
17563: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
17564: Procedure PError( Text : string)
17565: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17566: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
17567: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
17568: procedure playmp3(mpath: string);
17569: Procedure PlayMP31Click( Sender : TObject)
17570: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
17571: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
17572: procedure PolyBezier(const Points: array of TPoint);
17573: procedure PolyBezierTo(const Points: array of TPoint);
17574: procedure Polygon(const Points: array of TPoint);
17575: procedure Polyline(const Points: array of TPoint);
17576: Procedure Pop
17577: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
17578: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
17579: Procedure POPUP( X, Y : INTEGER)
17580: Procedure PopupURL(URL : WideString);
17581: Procedure POST
17582: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
17583: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
17584: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
17585: Procedure PostUser( const Email, FirstName, LastName : WideString)
17586: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
17587: procedure Pred(X: int64);
17588: Procedure Prepare
17589: Procedure PrepareStatement
17590: Procedure PreProcessXML( AList : TStrings)
17591: Procedure PreventDestruction
17592: Procedure Print( const Caption : string)
17593: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
17594: procedure printf(const format: String; const args: array of const);
17595: Procedure PrintList(Value: TStringList);
17596: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
17597: Procedure Printout1Click( Sender : TObject)
17598: Procedure ProcessHeaders
17599: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
17600: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
17601: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
17602: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
17603: Procedure ProcessMessagesOFF;  //application.processmessages
17604: Procedure ProcessMessagesON;
17605: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
17606: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
17607: Procedure Proclist Size is: 3797 /1415
17608: Procedure procMessClick( Sender : TObject)
17609: Procedure PSScriptCompile( Sender : TPSScript)
17610: Procedure PSScriptExecute( Sender : TPSScript)
17611: Procedure PSScriptLine( Sender : TObject)
17612: Procedure Push( ABoundary : string)
17613: procedure PushItem(AItem: Pointer)
17614: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
17615: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
17616: procedure PutLinuxLines(const Value: string)
17617: Procedure Quit
17618: Procedure RaiseConversionError( const AText : string);
17619: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
17620: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
17621: procedure RaiseException(Ex: TIFException; Param: String);
17622: Procedure RaiseExceptionForLastCmdResult;
17623: procedure RaiseLastException;
17624: procedure RaiseException2;
17625: Procedure RaiseLastOSError
17626: Procedure RaiseLastWin32;
17627: procedure RaiseLastWin32Error)
17628: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
17629: Procedure RandomFillStream( Stream : TMemoryStream)
17630: procedure randomize;
17631: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
```

```
17632: Procedure RCS
17633: Procedure Read( Socket : TSocket)
17634: Procedure ReadBlobData
17635: procedure ReadBuffer(Buffer:String;Count:LongInt)
17636: procedure ReadOnly1Click(Sender: TObject);
17637: Procedure ReadSection( const Section : string; Strings : TStrings)
17638: Procedure ReadSections( Strings : TStrings)
17639: Procedure ReadSections( Strings : TStrings);
17640: Procedure ReadSections1( const Section : string; Strings : TStrings);
17641: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
17642: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
17643: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
17644: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
17645: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
17646: Procedure Realign;
17647: procedure Rectangle(X1, Y1, X2, Y2: Integer);
17648: Procedure Rectangle1( const Rect : TRect);
17649: Procedure RectCopy( var Dest : TRect; const Source : TRect)
17650: Procedure RectFitToScreen( var R : TRect)
17651: Procedure RectGrow( var R : TRect; const Delta : Integer)
17652: Procedure RectGrowX( var R : TRect; const Delta : Integer)
17653: Procedure RectGrowY( var R : TRect; const Delta : Integer)
17654: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
17655: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
17656: Procedure RectNormalize( var R : TRect)
17657: //  TFileCallbackProcedure = procedure(filename:string);
17658: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
17659: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
17660: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
17661: Procedure Refresh;
17662: Procedure RefreshData( Options : TFetchOptions)
17663: Procedure REFRESHLOOKUPLIST
17664: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
17665: Procedure RegisterChanges( Value : TChangeLink)
17666: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
17667: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
17668: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
17669: Procedure ReInitialize( ADelay : Cardinal)
17670: procedure RELEASE
17671: Procedure Remove( const AByteCount : integer)
17672: Procedure REMOVE( FIELD : TFIELD)
17673: Procedure REMOVE( ITEM : TMENUITEM)
17674: Procedure REMOVE( POPUP : TPOPUPMENU)
17675: Procedure RemoveAllPasswords
17676: procedure RemoveComponent(AComponent:TComponent)
17677: Procedure RemoveDir( const ADirName : string)
17678: Procedure RemoveLambdaTransitions( var bChanged : boolean)
17679: Procedure REMOVEPARAM( VALUE : TPARAM)
17680: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
17681: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
17682: Procedure Rename( const ASourceFile, ADestFile : string)
17683: Procedure Rename( const FileName : string; Reload : Boolean)
17684: Procedure RenameTable( const NewTableName : string)
17685: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
17686: Procedure Replace1Click( Sender : TObject)
17687: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
17688: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
17689: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
17690: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
17691: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
17692: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
17693: Procedure Requery( Options : TExecuteOptions)
17694: Procedure Reset
17695: Procedure Reset1Click( Sender : TObject)
17696: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
17697: procedure ResourceExplore1Click(Sender: TObject);
17698: Procedure RestoreContents
17699: Procedure RestoreDefaults
17700: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
17701: Procedure RetrieveHeaders
17702: Procedure RevertRecord
17703: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
17704: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17705: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17706: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
17707: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
17708: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
17709: Procedure RleCompress2( Stream : TStream)
17710: Procedure RleDecompress2( Stream : TStream)
17711: Procedure RmDir(const S: string)
17712: Procedure Rollback
17713: Procedure Rollback( TransDesc : TTransactionDesc)
17714: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
17715: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
17716: Procedure RollbackTrans
17717: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
17718: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
17719: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
17720: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
```

```
17721: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
17722: Procedure S_EBox( const AText : string)
17723: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
17724: Procedure S_IBox( const AText : string)
17725: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
17726: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
17727: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
17728: Procedure SampleVarianceAndMean
17729: ( const X : TDynFloatArray; var Variance, Mean : Float)
17730: Procedure Save2Click( Sender : TObject)
17731: Procedure Saveas3Click( Sender : TObject)
17732: Procedure Savebefore1Click( Sender : TObject)
17733: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
17734: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
17735: Procedure SaveConfigFile
17736: Procedure SaveOutput1Click( Sender : TObject)
17737: procedure SaveScreenshotClick(Sender: TObject);
17738: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
17739: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
17740: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
17741: Procedure SaveToFile( AFileName : string)
17742: Procedure SAVETOFILE( const FILENAME : String)
17743: Procedure SaveToFile( const FileName : WideString)
17744: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
17745: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
17746: procedure SaveToFile(FileName: string);
17747: procedure SaveToFile(FileName:String)
17748: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
17749: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
17750: Procedure SaveToStream( S : TStream)
17751: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
17752: Procedure SaveToStream( Stream : TStream)
17753: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
17754: procedure SaveToStream(Stream: TStream);
17755: procedure SaveToStream(Stream:TStream)
17756: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
17757: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
17758: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
17759: procedure Say(const sText: string)
17760: Procedure SBytecode1Click( Sender : TObject)
17761: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
17762: procedure ScriptExplorer1Click(Sender: TObject);
17763: Procedure Scroll( Distance : Integer)
17764: Procedure Scroll( DX, DY : Integer)
17765: procedure ScrollBy(DeltaX, DeltaY: Integer);
17766: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
17767: Procedure ScrollTabs( Delta : Integer)
17768: Procedure Search1Click( Sender : TObject)
17769: procedure SearchAndOpenDoc(vfilenamepath: string)
17770: procedure SearchAndOpenFile(vfilenamepath: string)
17771: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
17772: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
17773: Procedure SearchNext1Click( Sender : TObject)
17774: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
17775: Procedure Select1( const Nodes : array of TTreeNode);
17776: Procedure Select2( Nodes : TList);
17777: Procedure SelectNext( Direction : Boolean)
17778: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
17779: Procedure SelfTestPEM  //unit uPSI_cPEM
17780: Procedure Send( AMsg : TIdMessage)
17781: //config forst in const MAILINIFILE = 'maildef.ini';
17782: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
17783: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
17784: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
17785: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
17786: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
17787: Procedure SendResponse
17788: Procedure SendStream( AStream : TStream)
17789: Procedure Set8087CW( NewCW : Word)
17790: Procedure SetAll( One, Two, Three, Four : Byte)
17791: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
17792: Procedure SetAppDispatcher( const ADispatcher : TComponent)
17793: procedure SetArrayLength;
17794: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
17795: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
17796: Procedure SetAsHandle( Format : Word; Value : THandle)
17797: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
17798: procedure SetCaptureControl(Control: TControl);
17799: Procedure SetColumnAttributes
17800: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
17801: Procedure SetCustomHeader( const Name, Value : string)
17802: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
17803: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
17804: Procedure SetFocus
17805: procedure SetFocus; virtual;
17806: Procedure SetInitialState
17807: Procedure SetKey
```

```
17808: procedure SetLastError(ErrorCode: Integer)
17809: procedure SetLength;
17810: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
17811: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
17812: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
17813: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
17814: Procedure SetParams1( UpdateKind : TUpdateKind);
17815: Procedure SetPassword( const Password : string)
17816: Procedure SetPointer( Ptr : Pointer; Size : Longint)
17817: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
17818: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
17819: Procedure SetProvider( Provider : TComponent)
17820: Procedure SetProxy( const Proxy : string)
17821: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
17822: Procedure SetRange( const StartValues, EndValues : array of const)
17823: Procedure SetRangeEnd
17824: Procedure SetRate( const aPercent, aYear : integer)
17825: procedure SetRate(const aPercent, aYear: integer)
17826: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
17827: Procedure SetSafeCallExceptionMsg( Msg : String)
17828: procedure SETSELTEXTBUF(BUFFER:PCHAR)
17829: Procedure SetSize( AWidth, AHeight : Integer)
17830: procedure SetSize(NewSize:LongInt)
17831: procedure SetString(var s: string; buffer: PChar; len: Integer)
17832: Procedure SetStrings( List : TStrings)
17833: Procedure SetText( Text : PwideChar)
17834: procedure SetText(Text: PChar);
17835: Procedure SetTextBuf( Buffer : PChar)
17836: procedure SETTEXTBUF(BUFFER:PCHAR)
17837: Procedure SetTick( Value : Integer)
17838: Procedure SetTimeout( ATimeOut : Integer)
17839: Procedure SetTraceEvent( Event : TDBXTraceEvent)
17840: Procedure SetUserName( const UserName : string)
17841: Procedure SetWallpaper( Path : string);
17842: Procedure ShellStyle1Click(Sender: TObject);
17843: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
17844: Procedure ShowFileProperties( const FileName : string)
17845: Procedure ShowInclude1Click( Sender : TObject)
17846: Procedure ShowInterfaces1Click( Sender : TObject)
17847: Procedure ShowLastException1Click( Sender : TObject)
17848: Procedure ShowMessage( const Msg : string)
17849: Procedure ShowMessageBig(const aText : string);
17850: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
17851: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
17852: Procedure MsgBig(const aText : string);        //alias
17853: procedure showmessage(mytext: string);
17854: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
17855: procedure ShowMessageFmt(const Msg: string; Params: array of const))
17856: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
17857: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
17858: Procedure ShowSearchDialog( const Directory : string)
17859: Procedure ShowSpecChars1Click( Sender : TObject)
17860: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
17861: Procedure ShredFile( const FileName : string; Times : Integer)
17862: procedure Shuffle(vQ: TStringList);
17863: Procedure ShuffleList( var List : array of Integer; Count : Integer)
17864: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
17865: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
17866: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
17867: Procedure Site( const ACommand : string)
17868: Procedure SkipEOL
17869: Procedure Sleep( ATime : cardinal)
17870: Procedure Sleep( milliseconds : Cardinal)
17871: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
17872: Procedure Slinenumbers1Click( Sender : TObject)
17873: Procedure Sort
17874: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
17875: procedure Speak(const sText: string)   //async like voice
17876: procedure Speak2(const sText: string)  //sync
17877: procedure Split(Str: string;  SubStr: string; List: TStrings);
17878: Procedure SplitNameValue( const Line : string; var Name, Value : string)
17879: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
17880: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
17881: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
17882: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
17883: procedure SQLSyntax1Click(Sender: TObject);
17884: Procedure SRand( Seed : RNG_IntType)
17885: Procedure Start
17886: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
17887: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
17888: Procedure StartTransaction( TransDesc : TTransactionDesc)
17889: Procedure Status( var AStatusList : TStringList)
17890: Procedure StatusBar1DblClick( Sender : TObject)
17891: Procedure StepInto1Click( Sender : TObject)
17892: Procedure StepIt
17893: Procedure StepOut1Click( Sender : TObject)
17894: Procedure Stop
17895: procedure stopmp3;
17896: procedure StartWeb(aurl: string);
```

```
17897: Procedure Str(aint: integer; astr: string); //of system
17898: Procedure StrDispose( Str : PChar)
17899: procedure StrDispose(Str: PChar)
17900: Procedure StrReplace(var Str: String; Old, New: String);
17901: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
17902: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
17903: Procedure StringToBytes( Value : String; Bytes : array of byte)
17904: procedure StrSet(c : Char; I : Integer; var s : String);
17905: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
17906: Procedure StructureMount( APath : String)
17907: procedure STYLECHANGED(SENDER:TOBJECT)
17908: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
17909: procedure Succ(X: int64);
17910: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
17911: procedure SwapChar(var X,Y: char);  //swapX follows
17912: Procedure SwapFloats( var X, Y : Float)
17913: procedure SwapGrid(grd: TStringGrid);
17914: Procedure SwapOrd( var I, J : Byte);
17915: Procedure SwapOrd( var X, Y : Integer)
17916: Procedure SwapOrd1( var I, J : Shortint);
17917: Procedure SwapOrd2( var I, J : Smallint);
17918: Procedure SwapOrd3( var I, J : Word);
17919: Procedure SwapOrd4( var I, J : Integer);
17920: Procedure SwapOrd5( var I, J : Cardinal);
17921: Procedure SwapOrd6( var I, J : Int64);
17922: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
17923: Procedure Synchronize1( Method : TMethod);
17924: procedure SyntaxCheck1Click(Sender: TObject);
17925: Procedure SysFreeString(const S: WideString); stdcall;
17926: Procedure TakeOver( Other : TLinearBitmap)
17927: Procedure Talkln(const sText: string)  //async voice
17928: procedure tbtn6resClick(Sender: TObject);
17929: Procedure tbtnUseCaseClick( Sender : TObject)
17930: procedure TerminalStyle1Click(Sender: TObject);
17931: Procedure Terminate
17932: Procedure texSyntax1Click( Sender : TObject)
17933: procedure TextOut(X, Y: Integer; Text: string);
17934: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
17935: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
17936: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
17937: Procedure TextStart
17938: procedure TILE
17939: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
17940: Procedure TitleClick( Column : TColumn)
17941: Procedure ToDo
17942: procedure toolbtnTutorialClick(Sender: TObject);
17943: Procedure Trace1( AURL : string; const AResponseContent : TStream);
17944: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
17945: Procedure Truncate
17946: procedure Tutorial101Click(Sender: TObject);
17947: procedure Tutorial10Statistics1Click(Sender: TObject);
17948: procedure Tutorial11Forms1Click(Sender: TObject);
17949: procedure Tutorial12SQL1Click(Sender: TObject);
17950: Procedure tutorial1Click( Sender : TObject)
17951: Procedure tutorial21Click( Sender : TObject)
17952: Procedure tutorial31Click( Sender : TObject)
17953: Procedure tutorial4Click( Sender : TObject)
17954: Procedure Tutorial5Click( Sender : TObject)
17955: procedure Tutorial6Click(Sender: TObject);
17956: procedure Tutorial91Click(Sender: TObject);
17957: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
17958: procedure UniqueString(var str: AnsiString)
17959: procedure UnloadLoadPackage(Module: HMODULE)
17960: Procedure Unlock
17961: Procedure UNMERGE( MENU : TMAINMENU)
17962: Procedure UnRegisterChanges( Value : TChangeLink)
17963: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
17964: Procedure UnregisterConversionType( const AType : TConvType)
17965: Procedure UnRegisterProvider( Prov : TCustomProvider)
17966: Procedure UPDATE
17967: Procedure UpdateBatch( AffectRecords : TAffectRecords)
17968: Procedure UPDATECURSORPOS
17969: Procedure UpdateFile
17970: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
17971: Procedure UpdateResponse( AResponse : TWebResponse)
17972: Procedure UpdateScrollBar
17973: Procedure UpdateView1Click( Sender : TObject)
17974: procedure Val(const s: string; var n, z: Integer)
17975: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
17976: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
17977: Procedure VariantAdd( const src : Variant; var dst : Variant)
17978: Procedure VariantAnd( const src : Variant; var dst : Variant)
17979: Procedure VariantArrayRedim( var V : Variant; High : Integer)
17980: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
17981: Procedure VariantClear( var V : Variant)
17982: Procedure VariantCpy( const src : Variant; var dst : Variant)
17983: Procedure VariantDiv( const src : Variant; var dst : Variant)
17984: Procedure VariantMod( const src : Variant; var dst : Variant)
17985: Procedure VariantMul( const src : Variant; var dst : Variant)
```

```
17986: Procedure VariantOr( const src : Variant; var dst : Variant)
17987: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
17988: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
17989: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
17990: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
17991: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
17992: Procedure VariantShl( const src : Variant; var dst : Variant)
17993: Procedure VariantShr( const src : Variant; var dst : Variant)
17994: Procedure VariantSub( const src : Variant; var dst : Variant)
17995: Procedure VariantXor( const src : Variant; var dst : Variant)
17996: Procedure VarCastError;
17997: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
17998: Procedure VarInvalidOp
17999: Procedure VarInvalidNullOp
18000: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
18001: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
18002: Procedure VarArrayCreateError
18003: Procedure VarResultCheck( AResult : HRESULT);
18004: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
18005: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
18006: Function VarTypeAsText( const AType : TVarType) : string
18007: procedure Voice(const sText: string) //async
18008: procedure Voice2(const sText: string) //sync
18009: Procedure WaitMiliSeconds( AMSec : word)
18010: Procedure WideAppend( var dst : WideString; const src : WideString)
18011: Procedure WideAssign( var dst : WideString; var src : WideString)
18012: Procedure WideDelete( var dst : WideString; index, count : Integer)
18013: Procedure WideFree( var s : WideString)
18014: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
18015: Procedure WideFromPChar( var dst : WideString; src : PChar)
18016: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
18017: Procedure WideSetLength( var dst : WideString; len : Integer)
18018: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
18019: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
18020: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
18021: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
18022: Procedure HttpGet(const Url: string; Stream:TStream);
18023: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
18024: Procedure WordWrap1Click( Sender : TObject)
18025: Procedure Write( const AOut : string)
18026: Procedure Write( Socket : TSocket)
18027: procedure Write(S: string);
18028: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
18029: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
18030: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
18031: procedure WriteBuffer(Buffer:String;Count:LongInt)
18032: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
18033: Procedure WriteChar( AValue : Char)
18034: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
18035: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
18036: Procedure WriteFloat( const Section, Name : string; Value : Double)
18037: Procedure WriteHeader( AHeader : TStrings)
18038: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
18039: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
18040: Procedure WriteLn( const AOut : string)
18041: procedure Writeln(s: string);
18042: Procedure WriteLog( const FileName, LogLine : string)
18043: Procedure WriteRFCReply( AReply : TIdRFCReply)
18044: Procedure WriteRFCStrings( AStrings : TStrings)
18045: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
18046: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
18047: Procedure WriteString( const Section, Ident, Value : String)
18048: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
18049: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
18050: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
18051: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18052: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18053: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
18054: procedure XMLSyntax1Click(Sender: TObject);
18055: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
18056: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
18057: Procedure ZeroFillStream( Stream : TMemoryStream)
18058: procedure XMLSyntax1Click(Sender: TObject);
18059: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
18060: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
18061: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
18062: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
18063: procedure(Sender, Source: TObject;X, Y: Integer)
18064: procedure(Sender, Target: TObject; X, Y: Integer)
18065: procedure(Sender: TObject; ASection, AWidth: Integer)
18066: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
18067: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
18068: procedure(Sender: TObject; var Action: TCloseAction)
18069: procedure(Sender: TObject; var CanClose: Boolean)
18070: procedure(Sender: TObject; var Key: Char);
18071: ProcedureName ProcedureNames ProcedureParametersCursor @
18072:
18073: *************Now Constructors constructor *************
18074:  Size is: 1248 1115 996 628 550 544 501 459 (381)
```

```
18075:  Attach( VersionInfoData : Pointer; Size : Integer)
18076:  constructor Create( ABuckets : TBucketListSizes)
18077:  Create( ACallBackWnd : HWND)
18078:  Create( AClient : TCustomTaskDialog)
18079:  Create( AClient : TIdTelnet)
18080:  Create( ACollection : TCollection)
18081:  Create( ACollection : TFavoriteLinkItems)
18082:  Create( ACollection : TTaskDialogButtons)
18083:  Create( AConnection : TIdCustomHTTP)
18084:  Create( ACreateSuspended : Boolean)
18085:  Create( ADataSet : TCustomSQLDataSet)
18086:  CREATE( ADATASET : TDATASET)
18087:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
18088:  Create( AGrid : TCustomDBGrid)
18089:  Create( AGrid : TStringGrid; AIndex : Longint)
18090:  Create( AHTTP : TIdCustomHTTP)
18091:  Create( AListItems : TListItems)
18092:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
18093:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
18094:  Create( AOwner : TCommonCalendar)
18095:  Create( AOwner : TComponent)
18096:  CREATE( AOWNER : TCOMPONENT)
18097:  Create( AOwner : TCustomListView)
18098:  Create( AOwner : TCustomOutline)
18099:  Create( AOwner : TCustomRichEdit)
18100:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
18101:  Create( AOwner : TCustomTreeView)
18102:  Create( AOwner : TIdUserManager)
18103:  Create( AOwner : TListItems)
18104:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
18105:  CREATE( AOWNER : TPERSISTENT)
18106:  Create( AOwner : TPersistent)
18107:  Create( AOwner : TTable)
18108:  Create( AOwner : TTreeNodes)
18109:  Create( AOwner : TWinControl; const ClassName : string)
18110:  Create( AParent : TIdCustomHTTP)
18111:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
18112:  Create( AProvider : TBaseProvider)
18113:  Create( AProvider : TCustomProvider);
18114:  Create( AProvider : TDataSetProvider)
18115:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
18116:  Create( ASocket : TSocket)
18117:  Create( AStrings : TWideStrings)
18118:  Create( AToolBar : TToolBar)
18119:  Create( ATreeNodes : TTreeNodes)
18120:  Create( Autofill : boolean)
18121:  Create( AWebPageInfo : TAbstractWebPageInfo)
18122:  Create( AWebRequest : TWebRequest)
18123:  Create( Collection : TCollection)
18124:  Create( Collection : TIdMessageParts; ABody : TStrings)
18125:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
18126:  Create( Column : TColumn)
18127:  Create( const AConvFamily : TConvFamily; const ADescription : string)
18128:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
18129:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
18130:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
18131:  Create( const ATabSet : TTabSet)
18132:  Create( const Compensate : Boolean)
18133:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
18134:  Create( const FileName : string)
18135:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
18136:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
18137:  Create( const MaskValue : string)
18138:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
18139:  Create( const Prefix : string)
18140:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
18141:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18142:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18143:  Create( CoolBar : TCoolBar)
18144:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
18145:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
18146:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
18147:  Create( DBCtrlGrid : TDBCtrlGrid)
18148:  Create( DSTableProducer : TDSTableProducer)
18149:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
18150:  Create( ErrorCode : DBIResult)
18151:  Create( Field : TBlobField; Mode : TBlobStreamMode)
18152:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
18153:  Create( HeaderControl : TCustomHeaderControl)
18154:  Create( HTTPRequest : TWebRequest)
18155:  Create( iStart : integer; sText : string)
18156:  Create( iValue : Integer)
18157:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
18158:  Create( MciErrNo : MCIERROR; const Msg : string)
18159:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
```

```
18160:  Create( Message : string; ErrorCode : DBResult)
18161:  Create( Msg : string)
18162:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
18163:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
18164:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
18165:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
18166:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
18167:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
18168:  Create( Owner : TCustomComboBoxEx)
18169:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
18170:  Create( Owner : TPersistent)
18171:  Create( Params : TStrings)
18172:  Create( Size : Cardinal)
18173:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
18174:  Create( StatusBar : TCustomStatusBar)
18175:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
18176:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
18177:  Create(AHandle:Integer)
18178:  Create(AOwner: TComponent); virtual;
18179:  Create(const AURI : string)
18180:  Create(FileName:String;Mode:Word)
18181:  Create(Instance:THandle;ResName:String;ResType:PChar)
18182:  Create(Stream : TStream)
18183:  Create1( ADataset : TDataset);
18184:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
18185:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
18186:  Create2( Other : TObject);
18187:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
18188:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
18189:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
18190:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
18191:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
18192:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
18193:  CreateRes( Ident : Integer);
18194:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
18195:  CreateRes( ResStringRec : PResStringRec);
18196:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
18197:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
18198:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
18199:  CreateSize( AWidth, AHeight : Integer)
18200:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
18201:
18202: ------------------------------------------------------------------------------
18203: unit uPSI_MathMax;
18204: ------------------------------------------------------------------------------
18205:  CONSTS
18206:   Bernstein: Float = 0.280169499023869133036436491230 7;  // Bernstein constant
18207:   Cbrt2: Float     = 1.259921049894873164767210607278 2;  // CubeRoot(2)
18208:   Cbrt3: Float     = 1.442249570307408382321638310780 1;  // CubeRoot(3)
18209:   Cbrt10: Float    = 2.154434690031883721759293566519 4;  // CubeRoot(10)
18210:   Cbrt100: Float   = 4.641588833612778892410076350919 4;  // CubeRoot(100)
18211:   CbrtPi: Float    = 1.464591887561523263020142527263 8;  // CubeRoot(PI)
18212:   Catalan: Float   = 0.915965594177219015054603514932 4;  // Catalan constant
18213:   PiJ:   Float       = 3.141592653589793238462643383279 5;  // PI
18214:   PI:  Extended =  3.14159265358979323846264338327 95);
18215:   PiOn2: Float     = 1.570796326794896619231321691639 8;  // PI / 2
18216:   PiOn3: Float     = 1.047197551196597746154214461093 2;  // PI / 3
18217:   PiOn4: Float     = 0.785398163397448309615660845819 88; // PI / 4
18218:   Sqrt2: Float     = 1.414213562373095048801688724209 7;  // Sqrt(2)
18219:   Sqrt3: Float     = 1.732050807568877293527446341505 9;  // Sqrt(3)
18220:   Sqrt5: Float     = 2.236067977499789696409173668731 3;  // Sqrt(5)
18221:   Sqrt10: Float    = 3.162277660168379331998893544432 7;  // Sqrt(10)
18222:   SqrtPi: Float    = 1.772453850905516027298167483341 1;  // Sqrt(PI)
18223:   Sqrt2Pi: Float   = 2.506628274631000502415765284811;   // Sqrt(2 * PI)
18224:   TwoPi: Float     = 6.283185307179586476925286766559;   // 2 * PI
18225:   ThreePi: Float   = 9.424777960769379715387930149838 5;  // 3 * PI
18226:   Ln2: Float       = 0.693147180559945309417232121458 18; // Ln(2)
18227:   Ln10: Float      = 2.302585092994045684017991454684 4;  // Ln(10)
18228:   LnPi: Float      = 1.144729885849400174143427351353 1;  // Ln(PI)
18229:   Log2J: Float     = 0.301029995663981195213738894724 49; // Log10(2)
18230:   Log3: Float      = 0.477121254719662437295027903255 12; // Log10(3)
18231:   LogPi: Float     = 0.497149872694133854351268288290 9;  // Log10(PI)
18232:   LogE: Float      = 0.434294481903251827651128918916 61; // Log10(E)
18233:   E: Float         = 2.718281828459045235360287471352 7;  // Natural constant
18234:   hLn2Pi: Float    = 0.918938533204672741780329736405 62; // Ln(2*PI)/2
18235:   inv2Pi: Float    = 0.159154943091895335768883763372514362034459645740 46; // 0.5/Pi
18236:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
18237:   GoldenMean: Float   = 1.618033988749894848204586834365638;  // GoldenMean
18238:   EulerMascheroni: Float = 0.577215664901532860606512090082 4;  // Euler GAMMA
18239:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
18240:   StDelta         : Extended = 0.00001;  {delta for difference equations}
18241:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
18242:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
18243:
18244: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
18245: begin
18246:   MetersPerInch = 0.0254; // [1]
18247:   MetersPerFoot = MetersPerInch * 12;
```

```
18248:   MetersPerYard = MetersPerFoot * 3;
18249:   MetersPerMile = MetersPerFoot * 5280;
18250:   MetersPerNauticalMiles = 1852;
18251:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
18252:   MetersPerLightSecond = 2.99792458E8; // [5]
18253:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
18254:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
18255:   MetersPerCubit = 0.4572; // [6][7]
18256:   MetersPerFathom = MetersPerFoot * 6;
18257:   MetersPerFurlong = MetersPerYard * 220;
18258:   MetersPerHand = MetersPerInch * 4;
18259:   MetersPerPace = MetersPerInch * 30;
18260:   MetersPerRod = MetersPerFoot * 16.5;
18261:   MetersPerChain = MetersPerRod * 4;
18262:   MetersPerLink = MetersPerChain / 100;
18263:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
18264:   MetersPerPica = MetersPerPoint * 12;
18265:
18266:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
18267:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
18268:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
18269:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
18270:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
18271:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
18272:
18273:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
18274:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
18275:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
18276:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
18277:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
18278:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
18279:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
18280:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
18281:
18282:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
18283:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
18284:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
18285:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
18286:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
18287:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
18288:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
18289:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
18290:
18291:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
18292:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
18293:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
18294:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
18295:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
18296:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
18297:
18298:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
18299:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
18300:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
18301:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
18302:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
18303:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
18304:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
18305:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
18306:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
18307:
18308:   GramsPerPound = 453.59237; // [1][7]
18309:   GramsPerDrams = GramsPerPound / 256;
18310:   GramsPerGrains = GramsPerPound / 7000;
18311:   GramsPerTons = GramsPerPound * 2000;
18312:   GramsPerLongTons = GramsPerPound * 2240;
18313:   GramsPerOunces = GramsPerPound / 16;
18314:   GramsPerStones = GramsPerPound * 14;
18315:
18316:   MaxAngle 9223372036854775808.0;
18317:   MaxTanH 5678.26170314707197474594655389854);
18318:   MaxFactorial( 1754);
18319:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
18320:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
18321:   MaxTanH( 354.89135644669199842162284618659);
18322:   MaxFactorial'LongInt'( 170);
18323:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
18324:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
18325:   MaxTanH( 44.36141955583649980270285577332 3);
18326:   MaxFactorial'LongInt'( 33);
18327:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
18328:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
18329:   PiExt( 3.14159265358979323846264338327 95);
18330:   RatioDegToRad( PiExt / 180.0);
18331:   RatioGradToRad( PiExt / 200.0);
18332:   RatioDegToGrad( 200.0 / 180.0);
18333:   RatioGradToDeg( 180.0 / 200.0);
18334:   Crc16PolynomCCITT'LongWord $1021);
18335:  Crc16PolynomIBM'LongWord $8005);
18336:  Crc16Bits'LongInt'( 16);
```

```
18337:  Crc16Bytes'LongInt'( 2);
18338:  Crc16HighBit'LongWord $8000);
18339:  NotCrc16HighBit','LongWord $7FFF);
18340:   Crc32PolynomIEEE','LongWord $04C11DB7);
18341:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
18342:  Crc32Koopman','LongWord $741B8CD7);
18343:  Crc32Bits','LongInt'( 32);
18344:  Crc32Bytes','LongInt'( 4);
18345:  Crc32HighBit','LongWord $80000000);
18346:  NotCrc32HighBit','LongWord $7FFFFFFF);
18347:
18348:   MinByte       = Low(Byte);
18349:   MaxByte       = High(Byte);
18350:   MinWord       = Low(Word);
18351:   MaxWord       = High(Word);
18352:   MinShortInt   = Low(ShortInt);
18353:   MaxShortInt   = High(ShortInt);
18354:   MinSmallInt   = Low(SmallInt);
18355:   MaxSmallInt   = High(SmallInt);
18356:   MinLongWord   = LongWord(Low(LongWord));
18357:   MaxLongWord   = LongWord(High(LongWord));
18358:   MinLongInt    = LongInt(Low(LongInt));
18359:   MaxLongInt    = LongInt(High(LongInt));
18360:   MinInt64      = Int64(Low(Int64));
18361:   MaxInt64      = Int64(High(Int64));
18362:   MinInteger    = Integer(Low(Integer));
18363:   MaxInteger    = Integer(High(Integer));
18364:   MinCardinal   = Cardinal(Low(Cardinal));
18365:   MaxCardinal   = Cardinal(High(Cardinal));
18366:   MinNativeUInt = NativeUInt(Low(NativeUInt));
18367:   MaxNativeUInt = NativeUInt(High(NativeUInt));
18368:   MinNativeInt  = NativeInt(Low(NativeInt));
18369:   MaxNativeInt  = NativeInt(High(NativeInt));
18370:  Function CosH( const Z : Float) : Float;
18371:  Function SinH( const Z : Float) : Float;
18372:  Function TanH( const Z : Float) : Float;
18373:
18374:
18375:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
18376:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
18377:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
18378:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
18379:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
18380:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
18381:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
18382:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
18383:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
18384:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
18385:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
18386:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
18387:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
18388:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
18389:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
18390:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
18391:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
18392:   MaxLog = 709.7827128933840;
18393:   MinLog = -708.3964185322641;
18394:   MaxFac = 170;
18395:   MaxGam = 171.624376956302;
18396:   MaxLgm = 2.556348E+305;
18397:   SingleCompareDelta   = 1.0E-34;
18398:   DoubleCompareDelta   = 1.0E-280;
18399:   {$IFDEF CLR}
18400:   ExtendedCompareDelta = DoubleCompareDelta;
18401:   {$ELSE}
18402:   ExtendedCompareDelta = 1.0E-4400;
18403:   {$ENDIF}
18404:   Bytes1KB  = 1024;
18405:   Bytes1MB  = 1024 * Bytes1KB;
18406:   Bytes1GB  = 1024 * Bytes1MB;
18407:   Bytes64KB = 64 * Bytes1KB;
18408:   Bytes64MB = 64 * Bytes1MB;
18409:   Bytes2GB  = 2 * LongWord(Bytes1GB);
18410:     clBlack32', $FF000000 ));
18411:     clDimGray32', $FF3F3F3F ));
18412:     clGray32', $FF7F7F7F ));
18413:     clLightGray32', $FFBFBFBF ));
18414:     clWhite32', $FFFFFFFF ));
18415:     clMaroon32', $FF7F0000 ));
18416:     clGreen32', $FF007F00 ));
18417:     clOlive32', $FF7F7F00 ));
18418:     clNavy32', $FF00007F ));
18419:     clPurple32', $FF7F007F ));
18420:     clTeal32', $FF007F7F ));
18421:     clRed32', $FFFF0000 ));
18422:     clLime32', $FF00FF00 ));
18423:     clYellow32', $FFFFFF00 ));
18424:     clBlue32', $FF0000FF ));
18425:     clFuchsia32', $FFFF00FF ));
```

```
18426:     clAqua32', $FF00FFFF ));
18427:     clAliceBlue32', $FFF0F8FF ));
18428:     clAntiqueWhite32', $FFFAEBD7 ));
18429:     clAquamarine32', $FF7FFFD4 ));
18430:     clAzure32', $FFF0FFFF ));
18431:     clBeige32', $FFF5F5DC ));
18432:     clBisque32', $FFFFE4C4 ));
18433:     clBlancheDalmond32', $FFFFEBCD ));
18434:     clBlueViolet32', $FF8A2BE2 ));
18435:     clBrown32', $FFA52A2A ));
18436:     clBurlyWood32', $FFDEB887 ));
18437:     clCadetblue32', $FF5F9EA0 ));
18438:     clChartReuse32', $FF7FFF00 ));
18439:     clChocolate32', $FFD2691E ));
18440:     clCoral32', $FFFF7F50 ));
18441:     clCornFlowerBlue32', $FF6495ED ));
18442:     clCornSilk32', $FFFFF8DC ));
18443:     clCrimson32', $FFDC143C ));
18444:     clDarkBlue32', $FF00008B ));
18445:     clDarkCyan32', $FF008B8B ));
18446:     clDarkGoldenRod32', $FFB8860B ));
18447:     clDarkGray32', $FFA9A9A9 ));
18448:     clDarkGreen32', $FF006400 ));
18449:     clDarkGrey32', $FFA9A9A9 ));
18450:     clDarkKhaki32', $FFBDB76B ));
18451:     clDarkMagenta32', $FF8B008B ));
18452:     clDarkOliveGreen32', $FF556B2F ));
18453:     clDarkOrange32', $FFFF8C00 ));
18454:     clDarkOrchid32', $FF9932CC ));
18455:     clDarkRed32', $FF8B0000 ));
18456:     clDarkSalmon32', $FFE9967A ));
18457:     clDarkSeaGreen32', $FF8FBC8F ));
18458:     clDarkSlateBlue32', $FF483D8B ));
18459:     clDarkSlateGray32', $FF2F4F4F ));
18460:     clDarkSlateGrey32', $FF2F4F4F ));
18461:     clDarkTurquoise32', $FF00CED1 ));
18462:     clDarkViolet32', $FF9400D3 ));
18463:     clDeepPink32', $FFFF1493 ));
18464:     clDeepSkyBlue32', $FF00BFFF ));
18465:     clDodgerBlue32', $FF1E90FF ));
18466:     clFireBrick32', $FFB22222 ));
18467:     clFloralWhite32', $FFFFFAF0 ));
18468:     clGainsBoro32', $FFDCDCDC ));
18469:     clGhostWhite32', $FFF8F8FF ));
18470:     clGold32', $FFFFD700 ));
18471:     clGoldenRod32', $FFDAA520 ));
18472:     clGreenYellow32', $FFADFF2F ));
18473:     clGrey32', $FF808080 ));
18474:     clHoneyDew32', $FFF0FFF0 ));
18475:     clHotPink32', $FFFF69B4 ));
18476:     clIndianRed32', $FFCD5C5C ));
18477:     clIndigo32', $FF4B0082 ));
18478:     clIvory32', $FFFFFFF0 ));
18479:     clKhaki32', $FFF0E68C ));
18480:     clLavender32', $FFE6E6FA ));
18481:     clLavenderBlush32', $FFFFF0F5 ));
18482:     clLawnGreen32', $FF7CFC00 ));
18483:     clLemonChiffon32', $FFFFFACD ));
18484:     clLightBlue32', $FFADD8E6 ));
18485:     clLightCoral32', $FFF08080 ));
18486:     clLightCyan32', $FFE0FFFF ));
18487:     clLightGoldenRodYellow32', $FFFAFAD2 ));
18488:     clLightGreen32', $FF90EE90 ));
18489:     clLightGrey32', $FFD3D3D3 ));
18490:     clLightPink32', $FFFFB6C1 ));
18491:     clLightSalmon32', $FFFFA07A ));
18492:     clLightSeagreen32', $FF20B2AA ));
18493:     clLightSkyblue32', $FF87CEFA ));
18494:     clLightSlategray32', $FF778899 ));
18495:     clLightSlategrey32', $FF778899 ));
18496:     clLightSteelblue32', $FFB0C4DE ));
18497:     clLightYellow32', $FFFFFFE0 ));
18498:     clLtGray32', $FFC0C0C0 ));
18499:     clMedGray32', $FFA0A0A4 ));
18500:     clDkGray32', $FF808080 ));
18501:     clMoneyGreen32', $FFC0DCC0 ));
18502:     clLegacySkyBlue32', $FFA6CAF0 ));
18503:     clCream32', $FFFFFBF0 ));
18504:     clLimeGreen32', $FF32CD32 ));
18505:     clLinen32', $FFFAF0E6 ));
18506:     clMediumAquamarine32', $FF66CDAA ));
18507:     clMediumBlue32', $FF0000CD ));
18508:     clMediumOrchid32', $FFBA55D3 ));
18509:     clMediumPurple32', $FF9370DB ));
18510:     clMediumSeaGreen32', $FF3CB371 ));
18511:     clMediumSlateBlue32', $FF7B68EE ));
18512:     clMediumSpringGreen32', $FF00FA9A ));
18513:     clMediumTurquoise32', $FF48D1CC ));
18514:     clMediumVioletRed32', $FFC71585 ));
```

```
18515:     clMidnightBlue32', $FF191970 ));
18516:     clMintCream32', $FFF5FFFA ));
18517:     clMistyRose32', $FFFFE4E1 ));
18518:     clMoccasin32', $FFFFE4B5 ));
18519:     clNavajoWhite32', $FFFFDEAD ));
18520:     clOldLace32', $FFFDF5E6 ));
18521:     clOliveDrab32', $FF6B8E23 ));
18522:     clOrange32', $FFFFA500 ));
18523:     clOrangeRed32', $FFFF4500 ));
18524:     clOrchid32', $FFDA70D6 ));
18525:     clPaleGoldenRod32', $FFEEE8AA ));
18526:     clPaleGreen32', $FF98FB98 ));
18527:     clPaleTurquoise32', $FFAFEEEE ));
18528:     clPaleVioletred32', $FFDB7093 ));
18529:     clPapayaWhip32', $FFFFEFD5 ));
18530:     clPeachPuff32', $FFFFDAB9 ));
18531:     clPeru32', $FFCD853F ));
18532:     clPlum32', $FFDDA0DD ));
18533:     clPowderBlue32', $FFB0E0E6 ));
18534:     clRosyBrown32', $FFBC8F8F ));
18535:     clRoyalBlue32', $FF4169E1 ));
18536:     clSaddleBrown32', $FF8B4513 ));
18537:     clSalmon32', $FFFA8072 ));
18538:     clSandyBrown32', $FFF4A460 ));
18539:     clSeaGreen32', $FF2E8B57 ));
18540:     clSeaShell32', $FFFFF5EE ));
18541:     clSienna32', $FFA0522D ));
18542:     clSilver32', $FFC0C0C0 ));
18543:     clSkyblue32', $FF87CEEB ));
18544:     clSlateBlue32', $FF6A5ACD ));
18545:     clSlateGray32', $FF708090 ));
18546:     clSlateGrey32', $FF708090 ));
18547:     clSnow32', $FFFFFAFA ));
18548:     clSpringgreen32', $FF00FF7F ));
18549:     clSteelblue32', $FF4682B4 ));
18550:     clTan32', $FFD2B48C ));
18551:     clThistle32', $FFD8BFD8 ));
18552:     clTomato32', $FFFF6347 ));
18553:     clTurquoise32', $FF40E0D0 ));
18554:     clViolet32', $FFEE82EE ));
18555:     clWheat32', $FFF5DEB3 ));
18556:     clWhitesmoke32', $FFF5F5F5 ));
18557:     clYellowgreen32', $FF9ACD32 ));
18558:     clTrWhite32', $7FFFFFFF ));
18559:     clTrBlack32', $7F000000 ));
18560:     clTrRed32', $7FFF0000 ));
18561:     clTrGreen32', $7F00FF00 ));
18562:     clTrBlue32', $7F0000FF ));
18563:   // Fixed point math constants
18564:   FixedOne = $10000;  FixedHalf = $7FFF;
18565:   FixedPI  = Round(PI * FixedOne);
18566:   FixedToFloat = 1/FixedOne;
18567:
18568:  Special Types
18569: ************************************************
18570:   type Complex = record
18571:     X, Y : Float;
18572:   end;
18573:   type TVector     = array of Float;
18574:   TIntVector  = array of Integer;
18575:   TCompVector = array of Complex;
18576:   TBoolVector = array of Boolean;
18577:   TStrVector  = array of String;
18578:   TMatrix     = array of TVector;
18579:   TIntMatrix  = array of TIntVector;
18580:   TCompMatrix = array of TCompVector;
18581:   TBoolMatrix = array of TBoolVector;
18582:   TStrMatrix  = array of TStrVector;
18583:   TByteArray = array[0..32767] of byte; !
18584:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
18585:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
18586:   T2StringArray = array of array of string;
18587:   T2IntegerArray = array of array of integer;
18588:   AddTypeS('INT_PTR', 'Integer
18589:   AddTypeS('LONG_PTR', 'Integer
18590:   AddTypeS('UINT_PTR', 'Cardinal
18591:   AddTypeS('ULONG_PTR', 'Cardinal
18592:   AddTypeS('DWORD_PTR', 'ULONG_PTR
18593:   TIntegerDynArray', 'array of Integer
18594:   TCardinalDynArray', 'array of Cardinal
18595:   TWordDynArray', 'array of Word
18596:   TSmallIntDynArray', 'array of SmallInt
18597:   TByteDynArray', 'array of Byte
18598:   TShortIntDynArray', 'array of ShortInt
18599:   TInt64DynArray', 'array of Int64
18600:   TLongWordDynArray', 'array of LongWord
18601:   TSingleDynArray', 'array of Single
18602:   TDoubleDynArray', 'array of Double
18603:   TBooleanDynArray', 'array of Boolean
```

```
18604:   TStringDynArray', 'array of string
18605:   TWideStringDynArray', 'array of WideString
18606:   TDynByteArray     = array of Byte;
18607:   TDynShortintArray = array of Shortint;
18608:   TDynSmallintArray = array of Smallint;
18609:   TDynWordArray     = array of Word;
18610:   TDynIntegerArray  = array of Integer;
18611:   TDynLongintArray  = array of Longint;
18612:   TDynCardinalArray = array of Cardinal;
18613:   TDynInt64Array    = array of Int64;
18614:   TDynExtendedArray = array of Extended;
18615:   TDynDoubleArray   = array of Double;
18616:   TDynSingleArray   = array of Single;
18617:   TDynFloatArray    = array of Float;
18618:   TDynPointerArray  = array of Pointer;
18619:   TDynStringArray   = array of string;
18620:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
18621:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
18622:   TSynSearchOptions = set of TSynSearchOption;
18623:
18624:
18625:
18626: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
18627: ----------------------------------------------------------------------------
18628: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
18629: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
18630: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
18631: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
18632: function CheckStringSum(vstring: string): integer;
18633: function HexToInt(HexNum: string): LongInt;
18634: function IntToBin(Int: Integer): String;
18635: function BinToInt(Binary: String): Integer;
18636: function HexToBin(HexNum: string): string; external2
18637: function BinToHex(Binary: String): string;
18638: function IntToFloat(i: Integer): double;
18639: function AddThousandSeparator(S: string; myChr: Char): string;
18640: function Max3(const X,Y,Z: Integer): Integer;
18641: procedure Swap(var X,Y: char); // faster without inline
18642: procedure ReverseString(var S: String);
18643: function CharToHexStr(Value: Char): string;
18644: function CharToUniCode(Value: Char): string;
18645: function Hex2Dec(Value: Str002): Byte;
18646: function HexStrCodeToStr(Value: string): string;
18647: function HexToStr(i: integer; value: string): string;
18648: function UniCodeToStr(Value: string): string;
18649: function CRC16(statement: string): string;
18650: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
18651: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
18652: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
18653: Procedure ExecuteCommand(executeFile, paramstring: string);
18654: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18655: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
18656: procedure SearchAndOpenDoc(vfilenamepath: string);
18657: procedure ShowInterfaces(myFile: string);
18658: function Fact2(av: integer): extended;
18659: Function BoolToStr(B: Boolean): string;
18660: Function GCD(x, y : LongInt) : LongInt;
18661: function LCM(m,n: longint): longint;
18662: function GetASCII: string;
18663: function GetItemHeight(Font: TFont): Integer;
18664: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
18665: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
18666: function getHINSTANCE: longword;
18667: function getHMODULE: longword;
18668: function GetASCII: string;
18669: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
18670: function WordIsOk(const AWord: string; var VW: Word): boolean;
18671: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
18672: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
18673: function SafeStr(const s: string): string;
18674: function ExtractUrlPath(const FileName: string): string;
18675: function ExtractUrlName(const FileName: string): string;
18676: function IsInternet: boolean;
18677: function RotateLeft1Bit_u32( Value: uint32): uint32;
18678: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats : Boolean);
18679: procedure getEnvironmentInfo;
18680: procedure AntiFreeze;
18681: function GetCPUSpeed: Double;
18682: function IsVirtualPcGuest : Boolean;
18683: function IsVmWareGuest : Boolean;
18684: procedure StartSerialDialog;
18685: function IsWoW64: boolean;
18686: function IsWow64String(var s: string): Boolean;
18687: procedure StartThreadDemo;
18688: Function RGB(R,G,B: Byte): TColor;
18689: Function Sendln(amess: string): boolean;
18690: Procedure maXbox;
18691: Function AspectRatio(aWidth, aHeight: Integer): String;
```

```
18692: function wget(aURL, afile: string): boolean;
18693: procedure PrintList(Value: TStringList);
18694: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
18695: procedure getEnvironmentInfo;
18696: procedure AntiFreeze;
18697: function getBitmap(apath: string): TBitmap;
18698: procedure ShowMessageBig(const aText : string);
18699: function YesNoDialog(const ACaption, AMsg: string): boolean;
18700: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
18701: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
18702: //function myStrToBytes(const Value: String): TBytes;
18703: //function myBytesToStr(const Value: TBytes): String;
18704: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
18705: function getBitmap(apath: string): TBitmap;
18706: procedure ShowMessageBig(const aText : string);
18707: Function StrToBytes(const Value: String): TBytes;
18708: Function BytesToStr(const Value: TBytes): String;
18709: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
18710: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
18711: function FindInPaths(const fileName, paths : String) : String;
18712: procedure initHexArray(var hexn: THexArray);
18713: function josephusG(n,k: integer; var graphout: string): integer;
18714: function isPowerof2(num: int64): boolean;
18715: function powerOf2(exponent: integer): int64;
18716: function getBigPI: string;
18717: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
18718:  function GetASCIILine: string;
18719: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
18720:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
18721: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
18722: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
18723: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
18724: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
18725: function isKeypressed: boolean;
18726: function Keypress: boolean;
18727: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
18728: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
18729: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
18730: function GetOSName: string;
18731: function GetOSVersion: string;
18732: function GetOSNumber: string;
18733: function getEnvironmentString: string;
18734: procedure StrReplace(var Str: String; Old, New: String);
18735: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18736: function getTeamViewerID: string;
18737: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
18738: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
18739: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
18740: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
18741: function StartSocketService: Boolean;
18742: procedure StartSocketServiceForm;
18743: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
18744: function GetFileList1(apath: string): TStringlist;
18745: procedure LetFileList(FileList: TStringlist; apath: string);
18746: procedure StartWeb(aurl: string);
18747: function GetTodayFiles(startdir, amask: string): TStringlist;
18748: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
18749: function JavahashCode(val: string): Integer;
18750: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18751: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
18752: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
18753: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
18754: Procedure ConvertToGray(Cnv: TCanvas);
18755: function GetFileDate(aFile:string; aWithTime:Boolean):string;
18756: procedure ShowMemory;
18757: function ShowMemory2: string;
18758: function getHostIP: string;
18759: procedure ShowBitmap(bmap: TBitmap);
18760: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
18761: function CreateDBGridForm(dblist: TStringList): TListbox;
18762: function isService: boolean;
18763: function isApplication: boolean;
18764: function isTerminalSession: boolean;
18765:
18766:
18767: // News of 3.9.8 up
18768: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
18769: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
18770: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
18771: JvChart - TJvChart Component - 2009 Public
18772: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
18773: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
18774: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
18775: DMath DLL included incl. Demos
18776: Interface Navigator menu/View/Intf Navigator
18777: Unit Explorer menu/Debug/Units Explorer
18778: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
18779: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
```

```
18780: Script History to 9 Files WebServer light /Options/Addons/WebServer
18781: Full Text Finder, JVSimLogic Simulator Package
18782: Halt-Stop Program in Menu, WebServer2, Stop Event ,
18783: Conversion Routines, Prebuild Forms, CodeSearch
18784: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
18785: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
18786: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
18787: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
18788: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
18789: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
18790: IDE Reflection API, Session Service Shell S3
18791: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
18792: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
18793: arduino map() function, PMRandom Generator
18794: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
18795: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
18796: REST Test Lib, Multilang Component, Forth Interpreter
18797: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
18798: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
18799: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
18800: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
18801: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
18802: QRCode Service, add more CFunctions like CDateTime of Synapse
18803: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
18804: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
18805: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
18806: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
18807: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
18808: BOLD Package, Indy Package5, maTRIx. MATHEMAX
18809: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
18810: emax layers: system-package-component-unit-class-function-block
18811: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
18812: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
18813: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
18814: OpenGL Game Demo: ..Options/Add Ons/Reversi
18815: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
18816: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
18817: 7% performance gain (hot spot profiling)
18818: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
18819: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
18820: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
18821: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
18822:
18823: add routines in 3.9.7.5
18824: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
18825: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
18826: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
18827: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
18828: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
18829: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
18830: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
18831:
18832: ////////////////////////// TestUnits //////////////////////////
18833:   SelftestPEM;
18834:   SelfTestCFundamentUtils;
18835:   SelfTestCFileUtils;
18836:   SelfTestCDateTime;
18837:   SelfTestCTimer;
18838:   SelfTestCRandom;
18839:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
18840:              Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
18841:
18842:   // Note: There's no need for installing a client certificate in the
18843:   //       webbrowser. The server asks the webbrowser to send a certificate but
18844:   //       if nothing is installed the software will work because the server
18845:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
18846:   //
18847:   //       file: c_cacert.p12
18848:   //       password: c_cakey
18849:
18850:   TGraphicControl = class(TControl)
18851:   private
18852:     FCanvas: TCanvas;
18853:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
18854:   protected
18855:     procedure Paint; virtual;
18856:     property Canvas: TCanvas read FCanvas;
18857:   public
18858:     constructor Create(AOwner: TComponent); override;
18859:     destructor Destroy; override;
18860:   end;
18861:
18862:   TCustomControl = class(TWinControl)
18863:   private
18864:     FCanvas: TCanvas;
18865:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
18866:   protected
18867:     procedure Paint; virtual;
18868:     procedure PaintWindow(DC: HDC); override;
```

```
18869:     property Canvas: TCanvas read FCanvas;
18870:   public
18871:     constructor Create(AOwner: TComponent); override;
18872:     destructor Destroy; override;
18873:   end;
18874:     RegisterPublishedProperties;
18875:     ('ONCHANGE', 'TNotifyEvent', iptrw);
18876:     ('ONCLICK', 'TNotifyEvent', iptrw);
18877:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
18878:     ('ONENTER', 'TNotifyEvent', iptrw);
18879:     ('ONEXIT', 'TNotifyEvent', iptrw);
18880:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
18881:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
18882:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
18883:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
18884:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
18885: //*********************************************************************
18886:   // To stop the while loop, click on Options/Show Include (boolean switch)!
18887:   Control a loop in a script with a form event:
18888:   IncludeON;   //control the while loop
18889:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
18890:
18891: //---------------------------------------------------------------------------
18892: //**************mX4 ini-file Configuration************************************
18893: //---------------------------------------------------------------------------
18894:   using config file maxboxdef.ini       menu/Help/Config File
18895:
18896: //*** Definitions for maXbox mX3 ***
18897: [FORM]
18898: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
18899: FONTSIZE=14
18900: EXTENSION=txt
18901: SCREENX=1386
18902: SCREENY=1077
18903: MEMHEIGHT=350
18904: PRINTFONT=Courier New //GUI Settings
18905: LINENUMBERS=Y   //line numbers at gutter in editor at left side
18906: EXCEPTIONLOG=Y  //store excepts and success in 2 log files see below! –menu Debug/Show Last Exceptions
18907: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
18908: BOOTSCRIPT=Y    //enabling load a boot script
18909: MEMORYREPORT=Y  //shows memory report on closing maXbox
18910: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
18911: NAVIGATOR=N     //shows function list at the right side of editor
18912: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
18913: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
18914: [WEB]
18915: IPPORT=8080     //for internal webserver – menu /Options/Add Ons/WebServer
18916: IPHOST=192.168.1.53
18917: ROOTCERT=filepathY
18918: SCERT=filepathY
18919: RSAKEY=filepathY
18920: VERSIONCHECK=Y
18921:
18922: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
18923:
18924: Also possible to set report memory in script to override ini setting
18925: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
18926:
18927:
18928: After Change the ini file you can reload the file with ../Help/Config Update
18929:
18930: //---------------------------------------------------------------------------
18931: //**************mX4 maildef.ini ini-file Configuration*********************
18932: //---------------------------------------------------------------------------
18933: //*** Definitions for maXMail ***
18934: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
18935: [MAXMAIL]
18936: HOST=getmail.softwareschule.ch
18937: USER=mailusername
18938: PASS=password
18939: PORT=110
18940: SSL=Y
18941: BODY=Y
18942: LAST=5
18943:
18944: ADO Connection String:
18945: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
18946:
18947: OpenSSL Lib:  unit ssl_openssl_lib;
18948: {$IFDEF CIL}
18949: const
18950:   {$IFDEF LINUX}
18951:   DLLSSLName = 'libssl.so';
18952:   DLLUtilName = 'libcrypto.so';
18953:   {$ELSE}
18954:   DLLSSLName = 'ssleay32.dll';
18955:   DLLUtilName = 'libeay32.dll';
18956:   {$ENDIF}
18957: {$ELSE}
```

```
18958: var
18959:   {$IFNDEF MSWINDOWS}
18960:     {$IFDEF DARWIN}
18961:     DLLSSLName: string = 'libssl.dylib';
18962:     DLLUtilName: string = 'libcrypto.dylib';
18963:     {$ELSE}
18964:     DLLSSLName: string = 'libssl.so';
18965:     DLLUtilName: string = 'libcrypto.so';
18966:     {$ENDIF}
18967:   {$ELSE}
18968:   DLLSSLName: string = 'ssleay32.dll';
18969:   DLLSSLName2: string = 'libssl32.dll';
18970:   DLLUtilName: string = 'libeay32.dll';
18971:   {$ENDIF}
18972: {$ENDIF}
18973:
18974:
18975: //------------------------------------------------------------------------
18976: //*************mX4 Macro Tags  ******************************************
18977: //------------------------------------------------------ ----------------
18978:
18979:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
      E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
18980:
18981: //Tag Macros
18982:
18983:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
18984:
18985: //Tag Macros
18986: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
18987: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
18988: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
18989: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
18990: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
18991: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
18992: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
18993: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
18994: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
18995:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
18996: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
18997: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
18998:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
18999: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
19000:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
19001: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
19002:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
19003:
19004: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
19005:
19006: //Replace Macros
19007:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
19008:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
19009:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
19010:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
19011:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
19012:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
19013:
19014:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
19015:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
19016: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
19017:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
19018:
19019: //------------------------------------------------------------------------
19020: //*************mX4 ToDo List Tags  ../Help/ToDo List**********************
19021: //------------------------------------------------------ ----------------
19022:
19023:     while I < sl.Count do begin
19024: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
19025:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
19026:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
19027:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
19028:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
19029:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
19030:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
19031:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
19032:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
19033:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
19034:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
19035:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
19036:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
19037:       Inc(I);
19038:     end;
19039:
19040:
19041: //------------------------------------------------------------------------
19042: //*************mX4 Public  Tools API ************************************
19043: //------------------------------------------------------------------------
19044:   file : unit uPSI_fMain.pas;               {$OTAP} Open Tools API Catalog
19045:  // Those functions concern the editor and preprocessor, all of the IDE
```

```
19046:  Example: Call it with maxform1.Info1Click(self)
19047:  Note: Call all Methods with maxForm1., e.g.:
19048:                        maxForm1.ShellStyle1Click(self);
19049:
19050: procedure SIRegister_fMain(CL: TPSPascalCompiler);
19051: begin
19052:  Const('BYTECODE','String 'bytecode.txt'
19053:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
19054:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
19055:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
19056:  Const('PSINC','String PS Includes (*.inc)|*.INC
19057:  Const('DEFFILENAME','String 'firstdemo.txt
19058:  Const('DEFINIFILE','String 'maxboxdef.ini
19059:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
19060:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
19061:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
19062:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
19063:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
19064:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
19065:  Const('INCLUDEBOX','String 'pas_includebox.inc
19066:  Const('BOOTSCRIPT','String 'maxbootscript.txt
19067:  Const('MBVERSION','String '3.9.9.96
19068:  Const('VERSION','String'3.9.9.96
19069:  Const('MBVER','String '399
19070:  Const('MBVERI','Integer'(399);
19071:  Const('MBVERIALL','Integer'(39996);
19072:  Const('EXENAME','String 'maXbox3.exe
19073:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
19074:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
19075:  Const('MXINTERNETCHECK','String 'www.ask.com
19076:  Const('MXMAIL','String 'max@kleiner.com
19077:  Const('TAB','Char #$09);
19078:  Const('CODECOMPLETION','String 'bds_delphi.dci
19079:  SIRegister_TMaxForm1(CL);
19080: end;
19081:
19082:   with FindClass('TForm'),'TMaxForm1') do begin
19083:     memo2', 'TMemo', iptrw);
19084:     memo1', 'TSynMemo', iptrw);
19085:     CB1SCList', 'TComboBox', iptrw);
19086:     mxNavigator', 'TComboBox', iptrw);
19087:     IPHost', 'string', iptrw);
19088:     IPPort', 'integer', iptrw);
19089:     COMPort', 'integer', iptrw);      //3.9.6.4
19090:     Splitter1', 'TSplitter', iptrw);
19091:     PSScript', 'TPSScript', iptrw);
19092:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
19093:     MainMenu1', 'TMainMenu', iptrw);
19094:     Program1', 'TMenuItem', iptrw);
19095:     Compile1', 'TMenuItem', iptrw);
19096:     Files1', 'TMenuItem', iptrw);
19097:     open1', 'TMenuItem', iptrw);
19098:     Save2', 'TMenuItem', iptrw);
19099:     Options1', 'TMenuItem', iptrw);
19100:     Savebefore1', 'TMenuItem', iptrw);
19101:     Largefont1', 'TMenuItem', iptrw);
19102:     sBytecode1', 'TMenuItem', iptrw);
19103:     Saveas3', 'TMenuItem', iptrw);
19104:     Clear1', 'TMenuItem', iptrw);
19105:     Slinenumbers1', 'TMenuItem', iptrw);
19106:     About1', 'TMenuItem', iptrw);
19107:     Search1', 'TMenuItem', iptrw);
19108:     SynPasSyn1', 'TSynPasSyn', iptrw);
19109:     memo1', 'TSynMemo', iptrw);
19110:     SynEditSearch1', 'TSynEditSearch', iptrw);
19111:     WordWrap1', 'TMenuItem', iptrw);
19112:     XPManifest1', 'TXPManifest', iptrw);
19113:     SearchNext1', 'TMenuItem', iptrw);
19114:     Replace1', 'TMenuItem', iptrw);
19115:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
19116:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
19117:     ShowInclude1', 'TMenuItem', iptrw);
19118:     SynEditPrint1', 'TSynEditPrint', iptrw);
19119:     Printout1', 'TMenuItem', iptrw);
19120:     mnPrintColors1', 'TMenuItem', iptrw);
19121:     dlgFilePrint', 'TPrintDialog', iptrw);
19122:     dlgPrintFont1', 'TFontDialog', iptrw);
19123:     mnuPrintFont1', 'TMenuItem', iptrw);
19124:     Include1', 'TMenuItem', iptrw);
19125:     CodeCompletionList1', 'TMenuItem', iptrw);
19126:     IncludeList1', 'TMenuItem', iptrw);
19127:     ImageList1', 'TImageList', iptrw);
19128:     ImageList2', 'TImageList', iptrw);
19129:     CoolBar1', 'TCoolBar', iptrw);
19130:     ToolBar1', 'TToolBar', iptrw);
19131:     tbtnLoad', 'TToolButton', iptrw);
19132:     ToolButton2', 'TToolButton', iptrw);
19133:     tbtnFind', 'TToolButton', iptrw);
19134:     tbtnCompile', 'TToolButton', iptrw);
```

```
19135:     tbtnTrans', 'TToolButton', iptrw);
19136:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
19137:     toolbtnTutorial', 'TToolButton', iptrw);
19138:     tbtn6res', 'TToolButton', iptrw);
19139:     ToolButton5', 'TToolButton', iptrw);
19140:     ToolButton1', 'TToolButton', iptrw);
19141:     ToolButton3', 'TToolButton', iptrw);
19142:     statusBar1', 'TStatusBar', iptrw);
19143:     SaveOutput1', 'TMenuItem', iptrw);
19144:     ExportClipboard1', 'TMenuItem', iptrw);
19145:     Close1', 'TMenuItem', iptrw);
19146:     Manual1', 'TMenuItem', iptrw);
19147:     About2', 'TMenuItem', iptrw);
19148:     loadLastfile1', 'TMenuItem', iptrw);
19149:     imglogo', 'TImage', iptrw);
19150:     cedebug', 'TPSScriptDebugger', iptrw);
19151:     debugPopupMenu1', 'TPopupMenu', iptrw);
19152:     BreakPointMenu', 'TMenuItem', iptrw);
19153:     Decompile1', 'TMenuItem', iptrw);
19154:     StepInto1', 'TMenuItem', iptrw);
19155:     StepOut1', 'TMenuItem', iptrw);
19156:     Reset1', 'TMenuItem', iptrw);
19157:     DebugRun1', 'TMenuItem', iptrw);
19158:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
19159:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
19160:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
19161:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
19162:     tutorial4', 'TMenuItem', iptrw);
19163:     ExporttoClipboard1', 'TMenuItem', iptrw);
19164:     ImportfromClipboard1', 'TMenuItem', iptrw);
19165:     N4', 'TMenuItem', iptrw);
19166:     N5', 'TMenuItem', iptrw);
19167:     N6', 'TMenuItem', iptrw);
19168:     ImportfromClipboard2', 'TMenuItem', iptrw);
19169:     tutorial1', 'TMenuItem', iptrw);
19170:     N7', 'TMenuItem', iptrw);
19171:     ShowSpecChars1', 'TMenuItem', iptrw);
19172:     OpenDirectory1', 'TMenuItem', iptrw);
19173:     procMess', 'TMenuItem', iptrw);
19174:     tbtnUseCase', 'TToolButton', iptrw);
19175:     ToolButton7', 'TToolButton', iptrw);
19176:     EditFont1', 'TMenuItem', iptrw);
19177:     UseCase1', 'TMenuItem', iptrw);
19178:     tutorial21', 'TMenuItem', iptrw);
19179:     OpenUseCase1', 'TMenuItem', iptrw);
19180:     PSImport_DB1', 'TPSImport_DB', iptrw);
19181:     tutorial31', 'TMenuItem', iptrw);
19182:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
19183:     HTMLSyntax1', 'TMenuItem', iptrw);
19184:     ShowInterfaces1', 'TMenuItem', iptrw);
19185:     Tutorial5', 'TMenuItem', iptrw);
19186:     AllFunctionsList1', 'TMenuItem', iptrw);
19187:     ShowLastException1', 'TMenuItem', iptrw);
19188:     PlayMP31', 'TMenuItem', iptrw);
19189:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
19190:     texSyntax1', 'TMenuItem', iptrw);
19191:     N8', 'TMenuItem', iptrw);
19192:     GetEMails1', 'TMenuItem', iptrw);
19193:     SynCppSyn1', 'TSynCppSyn', iptrw);
19194:     CSyntax1', 'TMenuItem', iptrw);
19195:     Tutorial6', 'TMenuItem', iptrw);
19196:     New1', 'TMenuItem', iptrw);
19197:     AllObjectsList1', 'TMenuItem', iptrw);
19198:     LoadBytecode1', 'TMenuItem', iptrw);
19199:     CipherFile1', 'TMenuItem', iptrw);
19200:     N9', 'TMenuItem', iptrw);
19201:     N10', 'TMenuItem', iptrw);
19202:     Tutorial11', 'TMenuItem', iptrw);
19203:     Tutorial71', 'TMenuItem', iptrw);
19204:     UpdateService1', 'TMenuItem', iptrw);
19205:     PascalSchool1', 'TMenuItem', iptrw);
19206:     Tutorial81', 'TMenuItem', iptrw);
19207:     DelphiSite1', 'TMenuItem', iptrw);
19208:     Output1', 'TMenuItem', iptrw);
19209:     TerminalStyle1', 'TMenuItem', iptrw);
19210:     ReadOnly1', 'TMenuItem', iptrw);
19211:     ShellStyle1', 'TMenuItem', iptrw);
19212:     BigScreen1', 'TMenuItem', iptrw);
19213:     Tutorial91', 'TMenuItem', iptrw);
19214:     SaveOutput2', 'TMenuItem', iptrw);
19215:     N11', 'TMenuItem', iptrw);
19216:     SaveScreenshot', 'TMenuItem', iptrw);
19217:     Tutorial101', 'TMenuItem', iptrw);
19218:     SQLSyntax1', 'TMenuItem', iptrw);
19219:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
19220:     Console1', 'TMenuItem', iptrw);
19221:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
19222:     XMLSyntax1', 'TMenuItem', iptrw);
19223:     ComponentCount1', 'TMenuItem', iptrw);
```

```
19224:     NewInstance1', 'TMenuItem', iptrw);
19225:     toolbtnTutorial', 'TToolButton', iptrw);
19226:     Memory1', 'TMenuItem', iptrw);
19227:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
19228:     JavaSyntax1', 'TMenuItem', iptrw);
19229:     SyntaxCheck1', 'TMenuItem', iptrw);
19230:     Tutorial10Statistics1', 'TMenuItem', iptrw);
19231:     ScriptExplorer1', 'TMenuItem', iptrw);
19232:     FormOutput1', 'TMenuItem', iptrw);
19233:     ArduinoDump1', 'TMenuItem', iptrw);
19234:     AndroidDump1', 'TMenuItem', iptrw);
19235:     GotoEnd1', 'TMenuItem', iptrw);
19236:     AllResourceList1', 'TMenuItem', iptrw);
19237:     ToolButton4', 'TToolButton', iptrw);
19238:     tbtn6res', 'TToolButton', iptrw);
19239:     Tutorial11Forms1', 'TMenuItem', iptrw);
19240:     Tutorial12SQL1', 'TMenuItem', iptrw);
19241:     ResourceExplore1', 'TMenuItem', iptrw);
19242:     Info1', 'TMenuItem', iptrw);
19243:     N12', 'TMenuItem', iptrw);
19244:     CryptoBox1', 'TMenuItem', iptrw);
19245:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
19246:     CipherFile2', 'TMenuItem', iptrw);
19247:     N13', 'TMenuItem', iptrw);
19248:     ModulesCount1', 'TMenuItem', iptrw);
19249:     AddOns2', 'TMenuItem', iptrw);
19250:     N4GewinntGame1', 'TMenuItem', iptrw);
19251:     DocuforAddOns1', 'TMenuItem', iptrw);
19252:     Tutorial14Async1', 'TMenuItem', iptrw);
19253:     Lessons15Review1', 'TMenuItem', iptrw);
19254:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
19255:     PHPSyntax1', 'TMenuItem', iptrw);
19256:     Breakpoint1', 'TMenuItem', iptrw);
19257:     SerialRS2321', 'TMenuItem', iptrw);
19258:     N14', 'TMenuItem', iptrw);
19259:     SynCSSyn1', 'TSynCSSyn', iptrw);
19260:     CSyntax2', 'TMenuItem', iptrw);
19261:     Calculator1', 'TMenuItem', iptrw);
19262:     tbtnSerial', 'TToolButton', iptrw);
19263:     ToolButton8', 'TToolButton', iptrw);
19264:     Tutorial151', 'TMenuItem', iptrw);
19265:     N15', 'TMenuItem', iptrw);
19266:     N16', 'TMenuItem', iptrw);
19267:     ControlBar1', 'TControlBar', iptrw);
19268:     ToolBar2', 'TToolBar', iptrw);
19269:     BtnOpen', 'TToolButton', iptrw);
19270:     BtnSave', 'TToolButton', iptrw);
19271:     BtnPrint', 'TToolButton', iptrw);
19272:     BtnColors', 'TToolButton', iptrw);
19273:     btnClassReport', 'TToolButton', iptrw);
19274:     BtnRotateRight', 'TToolButton', iptrw);
19275:     BtnFullSize', 'TToolButton', iptrw);
19276:     BtnFitToWindowSize', 'TToolButton', iptrw);
19277:     BtnZoomMinus', 'TToolButton', iptrw);
19278:     BtnZoomPlus', 'TToolButton', iptrw);
19279:     Panel1', 'TPanel', iptrw);
19280:     LabelBrettgroesse', 'TLabel', iptrw);
19281:     CB1SCList', 'TComboBox', iptrw);
19282:     ImageListNormal', 'TImageList', iptrw);
19283:     spbtnexplore', 'TSpeedButton', iptrw);
19284:     spbtnexample', 'TSpeedButton', iptrw);
19285:     spbsaveas', 'TSpeedButton', iptrw);
19286:     imglogobox', 'TImage', iptrw);
19287:     EnlargeFont1', 'TMenuItem', iptrw);
19288:     EnlargeFont2', 'TMenuItem', iptrw);
19289:     ShrinkFont1', 'TMenuItem', iptrw);
19290:     ThreadDemo1', 'TMenuItem', iptrw);
19291:     HEXEditor1', 'TMenuItem', iptrw);
19292:     HEXView1', 'TMenuItem', iptrw);
19293:     HEXInspect1', 'TMenuItem', iptrw);
19294:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
19295:     ExporttoHTML1', 'TMenuItem', iptrw);
19296:     ClassCount1', 'TMenuItem', iptrw);
19297:     HTMLOutput1', 'TMenuItem', iptrw);
19298:     HEXEditor2', 'TMenuItem', iptrw);
19299:     Minesweeper1', 'TMenuItem', iptrw);
19300:     N17', 'TMenuItem', iptrw);
19301:     PicturePuzzle1', 'TMenuItem', iptrw);
19302:     sbvclhelp', 'TSpeedButton', iptrw);
19303:     DependencyWalker1', 'TMenuItem', iptrw);
19304:     WebScanner1', 'TMenuItem', iptrw);
19305:     View1', 'TMenuItem', iptrw);
19306:     mnToolbar1', 'TMenuItem', iptrw);
19307:     mnStatusbar2', 'TMenuItem', iptrw);
19308:     mnConsole2', 'TMenuItem', iptrw);
19309:     mnCoolbar2', 'TMenuItem', iptrw);
19310:     mnSplitter2', 'TMenuItem', iptrw);
19311:     WebServer1', 'TMenuItem', iptrw);
19312:     Tutorial17Server1', 'TMenuItem', iptrw);
```

```
19313:     Tutorial18Arduino1', 'TMenuItem', iptrw);
19314:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
19315:     PerlSyntax1', 'TMenuItem', iptrw);
19316:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
19317:     PythonSyntax1', 'TMenuItem', iptrw);
19318:     DMathLibrary1', 'TMenuItem', iptrw);
19319:     IntfNavigator1', 'TMenuItem', iptrw);
19320:     EnlargeFontConsole1', 'TMenuItem', iptrw);
19321:     ShrinkFontConsole1', 'TMenuItem', iptrw);
19322:     SetInterfaceList1', 'TMenuItem', iptrw);
19323:     popintfList', 'TPopupMenu', iptrw);
19324:     intfAdd1', 'TMenuItem', iptrw);
19325:     intfDelete1', 'TMenuItem', iptrw);
19326:     intfRefactor1', 'TMenuItem', iptrw);
19327:     Defactor1', 'TMenuItem', iptrw);
19328:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
19329:     Tutorial20Regex', 'TMenuItem', iptrw);
19330:     N18', 'TMenuItem', iptrw);
19331:     ManualE1', 'TMenuItem', iptrw);
19332:     FullTextFinder1', 'TMenuItem', iptrw);
19333:     Move1', 'TMenuItem', iptrw);
19334:     FractalDemo1', 'TMenuItem', iptrw);
19335:     Tutorial21Android1', 'TMenuItem', iptrw);
19336:     Tutorial0Function1', 'TMenuItem', iptrw);
19337:     SimuLogBox1', 'TMenuItem', iptrw);
19338:     OpenExamples1', 'TMenuItem', iptrw);
19339:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
19340:     JavaScriptSyntax1', 'TMenuItem', iptrw);
19341:     Halt1', 'TMenuItem', iptrw);
19342:     CodeSearch1', 'TMenuItem', iptrw);
19343:     SynRubySyn1', 'TSynRubySyn', iptrw);
19344:     RubySyntax1', 'TMenuItem', iptrw);
19345:     Undo1', 'TMenuItem', iptrw);
19346:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
19347:     LinuxShellScript1', 'TMenuItem', iptrw);
19348:     Rename1', 'TMenuItem', iptrw);
19349:     spdcodesearch', 'TSpeedButton', iptrw);
19350:     Preview1', 'TMenuItem', iptrw);
19351:     Tutorial22Services1', 'TMenuItem', iptrw);
19352:     Tutorial23RealTime1', 'TMenuItem', iptrw);
19353:     Configuration1', 'TMenuItem', iptrw);
19354:     MP3Player1', 'TMenuItem', iptrw);
19355:     DLLSpy1', 'TMenuItem', iptrw);
19356:     SynURIOpener1', 'TSynURIOpener', iptrw);
19357:     SynURISyn1', 'TSynURISyn', iptrw);
19358:     URILinksClicks1', 'TMenuItem', iptrw);
19359:     EditReplace1', 'TMenuItem', iptrw);
19360:     GotoLine1', 'TMenuItem', iptrw);
19361:     ActiveLineColor1', 'TMenuItem', iptrw);
19362:     ConfigFile1', 'TMenuItem', iptrw);
19363:     Sort1Intflist', 'TMenuItem', iptrw);
19364:     Redo1', 'TMenuItem', iptrw);
19365:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
19366:     Tutorial25Configuration1', 'TMenuItem', iptrw);
19367:     IndentSelection1', 'TMenuItem', iptrw);
19368:     UnindentSection1', 'TMenuItem', iptrw);
19369:     SkyStyle1', 'TMenuItem', iptrw);
19370:     N19', 'TMenuItem', iptrw);
19371:     CountWords1', 'TMenuItem', iptrw);
19372:     imbookmarkimages', 'TImageList', iptrw);
19373:     Bookmark11', 'TMenuItem', iptrw);
19374:     N20', 'TMenuItem', iptrw);
19375:     Bookmark21', 'TMenuItem', iptrw);
19376:     Bookmark31', 'TMenuItem', iptrw);
19377:     Bookmark41', 'TMenuItem', iptrw);
19378:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
19379:
19380:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
19381:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
19382:     Procedure PSScriptCompile( Sender : TPSScript)
19383:     Procedure Compile1Click( Sender : TObject)
19384:     Procedure PSScriptExecute( Sender : TPSScript)
19385:     Procedure open1Click( Sender : TObject)
19386:     Procedure Save2Click( Sender : TObject)
19387:     Procedure Savebefore1Click( Sender : TObject)
19388:     Procedure Largefont1Click( Sender : TObject)
19389:     Procedure FormActivate( Sender : TObject)
19390:     Procedure SBytecode1Click( Sender : TObject)
19391:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
19392:     Procedure Saveas3Click( Sender : TObject)
19393:     Procedure Clear1Click( Sender : TObject)
19394:     Procedure Slinenumbers1Click( Sender : TObject)
19395:     Procedure About1Click( Sender : TObject)
19396:     Procedure Search1Click( Sender : TObject)
19397:     Procedure FormCreate( Sender : TObject)
19398:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
19399:                                                     var Action : TSynReplaceAction)
19400:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
19401:     Procedure WordWrap1Click( Sender : TObject)
```

```
19402:     Procedure SearchNext1Click( Sender : TObject)
19403:     Procedure Replace1Click( Sender : TObject)
19404:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
19405:     Procedure ShowInclude1Click( Sender : TObject)
19406:     Procedure Printout1Click( Sender : TObject)
19407:     Procedure mnuPrintFont1Click( Sender : TObject)
19408:     Procedure Include1Click( Sender : TObject)
19409:     Procedure FormDestroy( Sender : TObject)
19410:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
19411:     Procedure UpdateView1Click( Sender : TObject)
19412:     Procedure CodeCompletionList1Click( Sender : TObject)
19413:     Procedure SaveOutput1Click( Sender : TObject)
19414:     Procedure ExportClipboard1Click( Sender : TObject)
19415:     Procedure Close1Click( Sender : TObject)
19416:     Procedure Manual1Click( Sender : TObject)
19417:     Procedure LoadLastFile1Click( Sender : TObject)
19418:     Procedure Memo1Change( Sender : TObject)
19419:     Procedure Decompile1Click( Sender : TObject)
19420:     Procedure StepInto1Click( Sender : TObject)
19421:     Procedure StepOut1Click( Sender : TObject)
19422:     Procedure Reset1Click( Sender : TObject)
19423:     Procedure cedebugAfterExecute( Sender : TPSScript)
19424:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
19425:     Procedure cedebugCompile( Sender : TPSScript)
19426:     Procedure cedebugExecute( Sender : TPSScript)
19427:     Procedure cedebugIdle( Sender : TObject)
19428:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
19429:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
19430:     Procedure BreakPointMenuClick( Sender : TObject)
19431:     Procedure DebugRun1Click( Sender : TObject)
19432:     Procedure tutorial4Click( Sender : TObject)
19433:     Procedure ImportfromClipboard1Click( Sender : TObject)
19434:     Procedure ImportfromClipboard2Click( Sender : TObject)
19435:     Procedure tutorial1Click( Sender : TObject)
19436:     Procedure ShowSpecChars1Click( Sender : TObject)
19437:     Procedure StatusBar1DblClick( Sender : TObject)
19438:     Procedure PSScriptLine( Sender : TObject)
19439:     Procedure OpenDirectory1Click( Sender : TObject)
19440:     Procedure procMessClick( Sender : TObject)
19441:     Procedure tbtnUseCaseClick( Sender : TObject)
19442:     Procedure EditFont1Click( Sender : TObject)
19443:     Procedure tutorial21Click( Sender : TObject)
19444:     Procedure tutorial31Click( Sender : TObject)
19445:     Procedure HTMLSyntax1Click( Sender : TObject)
19446:     Procedure ShowInterfaces1Click( Sender : TObject)
19447:     Procedure Tutorial5Click( Sender : TObject)
19448:     Procedure ShowLastException1Click( Sender : TObject)
19449:     Procedure PlayMP31Click( Sender : TObject)
19450:     Procedure AllFunctionsList1Click( Sender : TObject)
19451:     Procedure texSyntax1Click( Sender : TObject)
19452:     Procedure GetEMails1Click( Sender : TObject)
19453:     procedure DelphiSite1Click(Sender: TObject);
19454:     procedure TerminalStyle1Click(Sender: TObject);
19455:     procedure ReadOnly1Click(Sender: TObject);
19456:     procedure ShellStyle1Click(Sender: TObject);
19457:     procedure Console1Click(Sender: TObject);     //3.2
19458:     procedure BigScreen1Click(Sender: TObject);
19459:     procedure Tutorial91Click(Sender: TObject);
19460:     procedure SaveScreenshotClick(Sender: TObject);
19461:     procedure Tutorial101Click(Sender: TObject);
19462:     procedure SQLSyntax1Click(Sender: TObject);
19463:     procedure XMLSyntax1Click(Sender: TObject);
19464:     procedure ComponentCount1Click(Sender: TObject);
19465:     procedure NewInstance1Click(Sender: TObject);
19466:     procedure CSyntax1Click(Sender: TObject);
19467:     procedure Tutorial6Click(Sender: TObject);
19468:     procedure New1Click(Sender: TObject);
19469:     procedure AllObjectsList1Click(Sender: TObject);
19470:     procedure LoadBytecode1Click(Sender: TObject);
19471:     procedure CipherFile1Click(Sender: TObject);  //V3.5
19472:     procedure NewInstance1Click(Sender: TObject);
19473:     procedure toolbtnTutorialClick(Sender: TObject);
19474:     procedure Memory1Click(Sender: TObject);
19475:     procedure JavaSyntax1Click(Sender: TObject);
19476:     procedure SyntaxCheck1Click(Sender: TObject);
19477:     procedure ScriptExplorer1Click(Sender: TObject);
19478:     procedure FormOutput1Click(Sender: TObject);  //V3.6
19479:     procedure GotoEnd1Click(Sender: TObject);
19480:     procedure AllResourceList1Click(Sender: TObject);
19481:     procedure tbtn6resClick(Sender: TObject);   //V3.7
19482:     procedure Info1Click(Sender: TObject);
19483:     procedure Tutorial10Statistics1Click(Sender: TObject);
19484:     procedure Tutorial11Forms1Click(Sender: TObject);
19485:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
19486:     procedure ResourceExplore1Click(Sender: TObject);
19487:     procedure Info1Click(Sender: TObject);
19488:     procedure CryptoBox1Click(Sender: TObject);
19489:     procedure ModulesCount1Click(Sender: TObject);
19490:     procedure N4GewinntGame1Click(Sender: TObject);
```

```
19491:     procedure PHPSyntax1Click(Sender: TObject);
19492:     procedure SerialRS2321Click(Sender: TObject);
19493:     procedure CSyntax2Click(Sender: TObject);
19494:     procedure Calculator1Click(Sender: TObject);
19495:     procedure Tutorial13Ciphering1Click(Sender: TObject);
19496:     procedure Tutorial14Async1Click(Sender: TObject);
19497:     procedure PHPSyntax1Click(Sender: TObject);
19498:     procedure BtnZoomPlusClick(Sender: TObject);
19499:     procedure BtnZoomMinusClick(Sender: TObject);
19500:     procedure btnClassReportClick(Sender: TObject);
19501:     procedure ThreadDemo1Click(Sender: TObject);
19502:     procedure HEXView1Click(Sender: TObject);
19503:     procedure ExporttoHTML1Click(Sender: TObject);
19504:     procedure Minesweeper1Click(Sender: TObject);
19505:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
19506:     procedure sbvclhelpClick(Sender: TObject);
19507:     procedure DependencyWalker1Click(Sender: TObject);
19508:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
19509:     procedure WebScanner1Click(Sender: TObject);
19510:     procedure mnToolbar1Click(Sender: TObject);
19511:     procedure mnStatusbar2Click(Sender: TObject);
19512:     procedure mnConsole2Click(Sender: TObject);
19513:     procedure mnCoolbar2Click(Sender: TObject);
19514:     procedure mnSplitter2Click(Sender: TObject);
19515:     procedure WebServer1Click(Sender: TObject);
19516:     procedure PerlSyntax1Click(Sender: TObject);
19517:     procedure PythonSyntax1Click(Sender: TObject);
19518:     procedure DMathLibrary1Click(Sender: TObject);
19519:     procedure IntfNavigator1Click(Sender: TObject);
19520:     procedure FullTextFinder1Click(Sender: TObject);
19521:     function AppName: string;
19522:     function ScriptName: string;
19523:     function LastName: string;
19524:     procedure FractalDemo1Click(Sender: TObject);
19525:     procedure SimuLogBox1Click(Sender: TObject);
19526:     procedure OpenExamples1Click(Sender: TObject);
19527:     procedure Halt1Click(Sender: TObject);
19528:     procedure Stop;
19529:     procedure CodeSearch1Click(Sender: TObject);
19530:     procedure RubySyntax1Click(Sender: TObject);
19531:     procedure Undo1Click(Sender: TObject);
19532:     procedure LinuxShellScript1Click(Sender: TObject);
19533:     procedure WebScannerDirect(urls: string);
19534:     procedure WebScanner(urls: string);
19535:     procedure LoadInterfaceList2;
19536:     procedure DLLSpy1Click(Sender: TObject);
19537:     procedure Memo1DblClick(Sender: TObject);
19538:     procedure URILinksClicks1Click(Sender: TObject);
19539:     procedure GotoLine1Click(Sender: TObject);
19540:     procedure ConfigFile1Click(Sender: TObject);
19541:     Procedure Sort1IntflistClick( Sender : TObject)
19542:     Procedure Redo1Click( Sender : TObject)
19543:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
19544:     Procedure IndentSelection1Click( Sender : TObject)
19545:     Procedure UnindentSection1Click( Sender : TObject)
19546:     Procedure SkyStyle1Click( Sender : TObject)
19547:     Procedure CountWords1Click( Sender : TObject)
19548:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
19549:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
19550:     Procedure Bookmark11Click( Sender : TObject)
19551:     Procedure Bookmark21Click( Sender : TObject)
19552:     Procedure Bookmark31Click( Sender : TObject)
19553:     Procedure Bookmark41Click( Sender : TObject)
19554:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
19555:     'STATMemoryReport', 'boolean', iptrw);
19556:     'IPPort', 'integer', iptrw);
19557:     'COMPort', 'integer', iptrw);
19558:     'lbintflist', 'TListBox', iptrw);
19559:     Function GetStatChange : boolean
19560:     Procedure SetStatChange( vstat : boolean)
19561:     Function GetActFileName : string
19562:     Procedure SetActFileName( vname : string)
19563:     Function GetLastFileName : string
19564:     Procedure SetLastFileName( vname : string)
19565:     Procedure WebScannerDirect( urls : string)
19566:     Procedure LoadInterfaceList2
19567:     Function GetStatExecuteShell : boolean
19568:     Procedure DoEditorExecuteCommand( EditorCommand : word)
19569:     function GetActiveLineColor: TColor
19570:     procedure SetActiveLineColor(acolor: TColor)
19571:     procedure ScriptListbox1Click(Sender: TObject);
19572:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
19573:     procedure EnlargeGutter1Click(Sender: TObject);
19574:     procedure Tetris1Click(Sender: TObject);
19575:     procedure ToDoList1Click(Sender: TObject);
19576:     procedure ProcessList1Click(Sender: TObject);
19577:     procedure MetricReport1Click(Sender: TObject);
19578:     procedure ProcessList1Click(Sender: TObject);
19579:     procedure TCPSockets1Click(Sender: TObject);
```

```
19580:     procedure ConfigUpdate1Click(Sender: TObject);
19581:     procedure ADOWorkbench1Click(Sender: TObject);
19582:     procedure SocketServer1Click(Sender: TObject);
19583:     procedure FormDemo1Click(Sender: TObject);
19584:     procedure Richedit1Click(Sender: TObject);
19585:     procedure SimpleBrowser1Click(Sender: TObject);
19586:     procedure DOSShell1Click(Sender: TObject);
19587:     procedure SynExport1Click(Sender: TObject);
19588:     procedure ExporttoRTF1Click(Sender: TObject);
19589:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
19590:     procedure SOAPTester1Click(Sender: TObject);
19591:     procedure Sniffer1Click(Sender: TObject);
19592:     procedure AutoDetectSyntax1Click(Sender: TObject);
19593:     procedure FPlot1Click(Sender: TObject);
19594:     procedure PasStyle1Click(Sender: TObject);
19595:     procedure Tutorial183RGBLED1Click(Sender: TObject);
19596:     procedure Reversi1Click(Sender: TObject);
19597:     procedure ManualmaXbox1Click(Sender: TObject);
19598:     procedure BlaisePascalMagazine1Click(Sender: TObject);
19599:     procedure AddToDo1Click(Sender: TObject);
19600:     procedure CreateGUID1Click(Sender: TObject);
19601:     procedure Tutorial27XML1Click(Sender: TObject);
19602:     procedure CreateDLLStub1Click(Sender: TObject);
19603:     procedure Tutorial28DLL1Click(Sender: TObject);');
19604:     procedure ResetKeyPressed;');
19605:     procedure FileChanges1Click(Sender: TObject);');
19606:     procedure OpenGLTry1Click(Sender: TObject);');
19607:     procedure AllUnitList1Click(Sender: TObject);');
19608:     procedure Tutorial29UMLClick(Sender: TObject);
19609:     procedure CreateHeader1Click(Sender: TObject);
19610:
19611: //-------------------------------------------------------------------------
19612: //*************mX4 Editor SynEdit Tools API ********************************
19613: //-------------------------------------------------------------------------
19614: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
19615: begin
19616:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
19617:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
19618:     Constructor Create( AOwner : TComponent)
19619:     SelStart', 'Integer', iptrw);
19620:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
19621:     Procedure UpdateCaret
19622:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
19623:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
19624:     Procedure BeginUndoBlock
19625:     Procedure BeginUpdate
19626:     Function CaretInView : Boolean
19627:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
19628:     Procedure Clear
19629:     Procedure ClearAll
19630:     Procedure ClearBookMark( BookMark : Integer)
19631:     Procedure ClearSelection
19632:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
19633:     Procedure ClearUndo
19634:     Procedure CopyToClipboard
19635:     Procedure CutToClipboard
19636:     Procedure DoCopyToClipboard( const SText : string)
19637:     Procedure EndUndoBlock
19638:     Procedure EndUpdate
19639:     Procedure EnsureCursorPosVisible
19640:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
19641:     Procedure FindMatchingBracket
19642:     Function GetMatchingBracket : TBufferCoord
19643:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
19644:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
19645:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
19646:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
19647:               : TSynHighlighterAttributes) : boolean
19648:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
19649:               var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
19650:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
19651:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
19652:     Procedure GotoBookMark( BookMark : Integer)
19653:     Procedure GotoLineAndCenter( ALine : Integer)
19654:     Function IdentChars : TSynIdentChars
19655:     Procedure InvalidateGutter
19656:     Procedure InvalidateGutterLine( aLine : integer)
19657:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
19658:     Procedure InvalidateLine( Line : integer)
19659:     Procedure InvalidateLines( FirstLine, LastLine : integer)
19660:     Procedure InvalidateSelection
19661:     Function IsBookmark( BookMark : integer) : boolean
19662:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
19663:     Procedure LockUndo
19664:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
19665:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
19666:     Function LineToRow( aLine : integer) : integer
19667:     Function RowToLine( aRow : integer) : integer
19668:     Function NextWordPos : TBufferCoord
```

```
19669:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
19670:     Procedure PasteFromClipboard
19671:     Function WordStart : TBufferCoord
19672:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
19673:     Function WordEnd : TBufferCoord
19674:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
19675:     Function PrevWordPos : TBufferCoord
19676:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
19677:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
19678:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
19679:     Procedure Redo
19680:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
19681:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
19682:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
19683:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
19684:     Procedure SelectAll
19685:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
19686:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
19687:     Procedure SetDefaultKeystrokes
19688:     Procedure SetSelWord
19689:     Procedure SetWordBlock( Value : TBufferCoord)
19690:     Procedure Undo
19691:     Procedure UnlockUndo
19692:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
19693:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
19694:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
19695:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
19696:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
19697:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
19698:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
19699:     Procedure AddFocusControl( aControl : TWinControl)
19700:     Procedure RemoveFocusControl( aControl : TWinControl)
19701:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
19702:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
19703:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
19704:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
19705:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
19706:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
19707:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
19708:     Procedure RemoveLinesPointer
19709:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
19710:     Procedure UnHookTextBuffer
19711:     BlockBegin', 'TBufferCoord', iptrw);
19712:     BlockEnd', 'TBufferCoord', iptrw);
19713:     CanPaste', 'Boolean', iptr);
19714:     CanRedo', 'boolean', iptr);
19715:     CanUndo', 'boolean', iptr);
19716:     CaretX', 'Integer', iptrw);
19717:     CaretY', 'Integer', iptrw);
19718:     CaretXY', 'TBufferCoord', iptrw);
19719:     ActiveLineColor', 'TColor', iptrw);
19720:     DisplayX', 'Integer', iptr);
19721:     DisplayY', 'Integer', iptr);
19722:     DisplayXY', 'TDisplayCoord', iptr);
19723:     DisplayLineCount', 'integer', iptr);
19724:     CharsInWindow', 'Integer', iptr);
19725:     CharWidth', 'integer', iptr);
19726:     Font', 'TFont', iptrw);
19727:     GutterWidth', 'Integer', iptr);
19728:     Highlighter', 'TSynCustomHighlighter', iptrw);
19729:     LeftChar', 'Integer', iptrw);
19730:     LineHeight', 'integer', iptr);
19731:     LinesInWindow', 'Integer', iptr);
19732:     LineText', 'string', iptrw);
19733:     Lines', 'TStrings', iptrw);
19734:     Marks', 'TSynEditMarkList', iptr);
19735:     MaxScrollWidth', 'integer', iptrw);
19736:     Modified', 'Boolean', iptrw);
19737:     PaintLock', 'Integer', iptr);
19738:     ReadOnly', 'Boolean', iptrw);
19739:     SearchEngine', 'TSynEditSearchCustom', iptrw);
19740:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
19741:     SelTabBlock', 'Boolean', iptr);
19742:     SelTabLine', 'Boolean', iptr);
19743:     SelText', 'string', iptrw);
19744:     StateFlags', 'TSynStateFlags', iptr);
19745:     Text', 'string', iptrw);
19746:     TopLine', 'Integer', iptrw);
19747:     WordAtCursor', 'string', iptr);
19748:     WordAtMouse', 'string', iptr);
19749:     UndoList', 'TSynEditUndoList', iptr);
19750:     RedoList', 'TSynEditUndoList', iptr);
19751:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
19752:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
19753:     BorderStyle', 'TSynBorderStyle', iptrw);
19754:     ExtraLineSpacing', 'integer', iptrw);
19755:     Gutter', 'TSynGutter', iptrw);
19756:     HideSelection', 'boolean', iptrw);
19757:     InsertCaret', 'TSynEditCaretType', iptrw);
```

```
19758:     InsertMode', 'boolean', iptrw);
19759:     IsScrolling', 'Boolean', iptr);
19760:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
19761:     MaxUndo', 'Integer', iptrw);
19762:     Options', 'TSynEditorOptions', iptrw);
19763:     OverwriteCaret', 'TSynEditCaretType', iptrw);
19764:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
19765:     ScrollHintColor', 'TColor', iptrw);
19766:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
19767:     ScrollBars', 'TScrollStyle', iptrw);
19768:     SelectedColor', 'TSynSelectedColor', iptrw);
19769:     SelectionMode', 'TSynSelectionMode', iptrw);
19770:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
19771:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
19772:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
19773:     WordWrapGlyph', 'TSynGlyph', iptrw);
19774:     OnChange', 'TNotifyEvent', iptrw);
19775:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
19776:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
19777:     OnContextHelp', 'TContextHelpEvent', iptrw);
19778:     OnDropFiles', 'TDropFilesEvent', iptrw);
19779:     OnGutterClick', 'TGutterClickEvent', iptrw);
19780:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
19781:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
19782:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
19783:     OnPaint', 'TPaintEvent', iptrw);
19784:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
19785:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
19786:     OnReplaceText', 'TReplaceTextEvent', iptrw);
19787:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
19788:     OnStatusChange', 'TStatusChangeEvent', iptrw);
19789:     OnPaintTransient', 'TPaintTransient', iptrw);
19790:     OnScroll', 'TScrollEvent', iptrw);
19791:   end;
19792:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
19793:  Function GetPlaceableHighlighters : TSynHighlighterList
19794:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
19795:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
19796:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
19797:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
19798:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
19799:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
19800:  Function ConvertCodeStringToExtended( AString : String) : String
19801:  Function ConvertExtendedToCodeString( AString : String) : String
19802:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
19803:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
19804:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
19805:
19806:   TSynEditorOption = (
19807:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
19808:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
19809:                                // preceding line
19810:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
19811:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
19812:                                //direction any more
19813:     eoDragDropEditing,         //Allows to select a block of text and drag it within document to another
19814:                                // location
19815:     eoDropFiles,               //Allows the editor accept OLE file drops
19816:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
19817:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
19818:     eoGroupUndo,               //When undoing/redoing actions, handle all continous changes the same kind
19819:                                // in one call
19820:                                //instead undoing/redoing each command separately
19821:     eoHalfPageScroll,          //When scrolling with page-up and page-down commands, only scroll a half
19822:                                //page at a time
19823:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
19824:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
19825:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
19826:     eoNoCaret,                 //Makes it so the caret is never visible
19827:     eoNoSelection,             //Disables selecting text
19828:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
19829:     eoScrollByOneLess,         //Forces scrolling to be one less
19830:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
19831:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
19832:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
19833:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
19834:     eoShowSpecialChars,        //Shows the special Characters
19835:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
19836:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
19837:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
19838:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
19839:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
19840:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
19841:
19842:     *********************************Important Editor Short Cuts*****************************);
19843:     Double click to select a word and count words with highlightning.
19844:     Triple click to select a line.
19845:     CTRL+SHIFT+click to extend a selection.
19846:     Drag with the ALT key down to select columns of text !!!
```

```
19847:     Drag and drop is supported.
19848:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
19849:     Type CTRL+A to select all.
19850:     Type CTRL+N to set a new line.
19851:     Type CTRL+T to delete a line or token. //Tokenizer
19852:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
19853:     Type CTRL+Shift+T to add ToDo in line and list.
19854:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
19855:     Type CTRL[0..9] to jump or get to bookmarks.
19856:     Type Home to position cursor at beginning of current line and End to position it at end of line.
19857:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
19858:     Page Up and Page Down work as expected.
19859:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
19860:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
19861:
19862: {$ Short Key Positions Ctrl<A-Z>: }
19863: def
19864:    <A> Select All
19865:    <B> Count Words
19866:    <C> Copy
19867:    <D> Internet Start
19868:    <E> Script List
19869:    <F> Find
19870:    <G> Goto
19871:    <H> Mark Line
19872:    <I> Interface List
19873:    <J> Code Completion
19874:    <K> Console
19875:    <L> Interface List Box
19876:    <M> Font Larger -
19877:    <N> New Line
19878:    <O> Open File
19879:    <P> Font Smaller +
19880:    <Q> Quit
19881:    <R> Replace
19882:    <S> Save!
19883:    <T> Delete Line
19884:    <U> Use Case Editor
19885:    <V> Paste
19886:    <W> URI Links
19887:    <X> Reserved for coding use internal
19888:    <Y> Delete Line
19889:    <Z> Undo
19890:
19891: ref
19892:    F1 Help
19893:    F2 Syntax Check
19894:    F3 Search Next
19895:    F4 New Instance
19896:    F5 Line Mark /Breakpoint
19897:    F6 Goto End
19898:    F7 Debug Step Into
19899:    F8 Debug Step Out
19900:    F9 Compile
19901:    F10 Menu
19902:    F11 Word Count Highlight
19903:    F12 Reserved for coding use internal
19904:
19905:  def ReservedWords: array[0..82] of string =
19906:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
19907:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
19908:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
19909:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
19910:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
19911:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
19912:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
19913:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
19914:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
19915:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
19916:      'public', 'published','def','ref','using','typedef','memo1','memo2','doc','maxform1';
19917:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
19918:
19919: //-------------------------------------------------------------------------
19920: //*************End of mX4 Public  Tools API *********************************
19921: //-------------------------------------------------------------------------
19922:
19923: Amount of Functions: 12682
19924: Amount of Procedures: 7858
19925: Amount of Constructors: 1275
19926: Totals of Calls: 21815
19927: SHA1: Win 3.9.9.96 EDD7FC051FF703851938AA33B4FBACCBA2552F1C
19928:
19929: ************************************************************
19930:  Doc Short Manual with 50 Tips!
19931: ************************************************************
19932: - Install: just save your maxboxdef.ini before and then extract the zip file!
19933: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
19934: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
19935: - Menu: With <Crtl><F3> you can search for code on examples
```

```
19936: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
19937: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
19938: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
19939:
19940: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
19941: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
19942: - Context Menu: You can printout your scripts as a pdf-file or html-export
19943: - Context: You do have a context menu with the right mouse click
19944:
19945: - Menu: With the UseCase Editor you can convert graphic formats too.
19946: - Menu: On menu Options you find Addons as compiled scripts
19947: - IDE: You don't need a mouse to handle maXbox, use shortcuts
19948: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
19949: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
19950: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
19951:         or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
19952:
19953: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
19954: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
19955: - Code: If you code a loop till key-pressed use function: isKeyPressed;
19956: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
19957: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
19958: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
19959:         to delete and Click and mark to drag a bookmark
19960: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
19961: - IDE: A file info with system and script information you find in menu Program/Information
19962: - IDE: After change the config file in help you can update changes in menu Help/Config Update
19963: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
19964: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
19965: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
19966: - Editor: Set Bookmarks to check your work in app or code
19967: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
19968: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
19969: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
19970:
19971: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
19972: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
19973: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
19974: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
19975: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
19976: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
19977: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
19978: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
19979:
19980: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
19981: - Add on when no browser is available start /Options/Add ons/Easy Browser
19982: - Add on SOAP Tester with SOP POST File
19983: - Add on IP Protocol Sniffer with List View
19984: - Add on OpenGL mX Robot Demo for android
19985:
19986: - Menu: Help/Tools as a Tool Section with DOS Opener
19987: - Menu Editor: export the code as RTF File
19988: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
19989: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
19990: - Context: Auto Detect of Syntax depending on file extension
19991: - Code: some Windows API function start with w in the name like wGetAtomName();
19992: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
19993: - IDE File Check with menu ..View/File Changes/...
19994: - Context: Create a Header with Create Header in Navigator List at right window
19995: - Code: use  SysErrorMessage to get a real Error Description, Ex.
19996:         RemoveDir('c:\NoSuchFolder') ;
19997:         writeln('System Error Message: '+ SysErrorMessage(GetLastError));
19998: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
19999:
20000:
20001: - using DLL example in maXbox: //function: {*****************************************}
20002:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
20003:                                  cb: DWORD): BOOL; //stdcall;;
20004:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
20005:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
20006:      External  'OpenProcess@kernel32.dll stdcall';
20007:
20008: PCT Precompile Technology , mX4 ScriptStudio
20009: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
20010: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
20011: emax layers: system-package-component-unit-class-function-block
20012: new keywords def ref using maXCalcF
20013: UML: use case act class state seq pac comp dep - lib lab
20014: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
20015: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
20016: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
20017: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
20018: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
20019: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
20020: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
20021: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
20022: GetScript or GetWebScript, GPS Example
20023:
20024:
```

```
20025: https://unibe-ch.academia.edu/MaxKleiner
20026: www.slideshare.net/maxkleiner1
20027: http://www.scribd.com/max_kleiner
20028: http://www.delphiforfun.org/Programs/Utilities/index.htm
20029: http://www.slideshare.net/maxkleiner1
20030: http://s3.amazonaws.com/PreviewLinks/22959.html
20031: http://www.softwareschule.ch/arduino_training.pdf
20032:
20033:
20034:
20035:
20036:
20037:
20038: ************************************************************
20039:  unit List asm internal end
20040: ************************************************************
20041: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
20042: 02 unit SIRegister_IdStrings                   //Indy Sockets
20043: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
20044: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
20045: 05 unit IFSI_WinForm1puzzle;                   //maXbox
20046: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
20047: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
20048: 08 unit RIRegister_MathMax_Routines(exec);    //Jedi & Delphi
20049: 09 unit RIRegister_IdGlobal_Routines(exec);   //Indy Sockets
20050: 10 unit RIRegister_SysUtils_Routines(Exec);   //Delphi
20051: 11 unit uPSI_IdTCPConnection;                 //Indy some functions
20052: 12 unit uPSCompiler.pas;                      //PS kernel functions
20053: 13 unit uPSI_DBCommon;                        //DB Common_Routines and Types
20054: 14 unit uPSI_Printers.pas                     //Delphi VCL
20055: 15 unit uPSI_MPlayer.pas                     //Delphi VCL
20056: 16 unit uPSC_comobj;                         //COM Functions
20057: 17 unit uPSI_Clipbrd;                        //Delphi VCL
20058: 18 unit Filectrl in IFSI_SysUtils_max;       //VCL Runtime
20059: 19 unit uPSI_SqlExpr;                       //DBX3
20060: 20 unit uPSI_ADODB;                         //ADODB
20061: 21 unit uPSI_StrHlpr;                       //String Helper Routines
20062: 22 unit uPSI_DateUtils;                     //Expansion to DateTimeLib
20063: 23 unit uPSI_FileUtils;                     //Expansion to Sys/File Utils
20064: 24 unit JUtils / gsUtils;                   //Jedi / Metabase
20065: 25 unit JvFunctions_max;                    //Jedi Functions
20066: 26 unit HTTPParser;                         //Delphi VCL
20067: 27 unit HTTPUtil;                          //Delphi VCL
20068: 28 unit uPSI_XMLUtil;                      //Delphi VCL
20069: 29 unit uPSI_SOAPHTTPClient;               //Delphi VCL SOAP WebService V3.5
20070: 30 unit uPSI_Contnrs;                      //Delphi RTL Container of Classes
20071: 31 unit uPSI_MaskUtils;                    //RTL Edit and Mask functions
20072: 32 unit uPSI_MyBigInt;                     //big integer class with Math
20073: 33 unit uPSI_ConvUtils;                    //Delphi VCL Conversions engine
20074: 34 unit Types_Variants;                    //Delphi\Win32\rtl\sys
20075: 35 unit uPSI_IdHashSHA1;                   //Indy Crypto Lib
20076: 36 unit uPSI_IdHashMessageDigest           //Indy Crypto;
20077: 37 unit uPSI_IdASN1Util;                   //Indy ASN1Utility Routines;
20078: 38 unit uPSI_IdLogFile;                    //Indy Logger from LogBase
20079: 39 unit uPSI_IdIcmpClient;                 //Indy Ping ICMP
20080: 40 unit uPSI_IdHashMessageDigest_max       //Indy Crypto &OpenSSL;
20081: 41 unit uPSI_FileCtrl;                     //Delphi RTL
20082: 42 unit uPSI_Outline;                      //Delphi VCL
20083: 43 unit uPSI_ScktComp;                     //Delphi RTL
20084: 44 unit uPSI_Calendar;                     //Delphi VCL
20085: 45 unit uPSI_VListView                     //VListView;
20086: 46 unit uPSI_DBGrids;                      //Delphi VCL
20087: 47 unit uPSI_DBCtrls;                      //Delphi VCL
20088: 48 unit ide_debugoutput;                   //maXbox
20089: 49 unit uPSI_ComCtrls;                     //Delphi VCL
20090: 50 unit uPSC_stdctrls+;                    //Delphi VCL
20091: 51 unit uPSI_Dialogs;                      //Delphi VCL
20092: 52 unit uPSI_StdConvs;                     //Delphi RTL
20093: 53 unit uPSI_DBClient;                     //Delphi RTL
20094: 54 unit uPSI_DBPlatform;                   //Delphi RTL
20095: 55 unit uPSI_Provider;                     //Delphi RTL
20096: 56 unit uPSI_FMTBcd;                       //Delphi RTL
20097: 57 unit uPSI_DBCGrids;                     //Delphi VCL
20098: 58 unit uPSI_CDSUtil;                      //MIDAS
20099: 59 unit uPSI_VarHlpr;                      //Delphi RTL
20100: 60 unit uPSI_ExtDlgs;                      //Delphi VCL
20101: 61 unit sdpStopwatch;                      //maXbox
20102: 62 unit uPSI_JclStatistics;                //JCL
20103: 63 unit uPSI_JclLogic;                     //JCL
20104: 64 unit uPSI_JclMiscel;                    //JCL
20105: 65 unit uPSI_JclMath_max;                  //JCL RTL
20106: 66 unit uPSI_uTPLb_StreamUtils;            //LockBox 3
20107: 67 unit uPSI_MathUtils;                    //BCB
20108: 68 unit uPSI_JclMultimedia;                //JCL
20109: 69 unit uPSI_WideStrUtils;                 //Delphi API/RTL
20110: 70 unit uPSI_GraphUtil;                    //Delphi RTL
20111: 71 unit uPSI_TypeTrans;                    //Delphi RTL
20112: 72 unit uPSI_HTTPApp;                      //Delphi VCL
20113: 73 unit uPSI_DBWeb;                        //Delphi VCL
```

```
20114: 74 unit uPSI_DBBdeWeb;                      //Delphi VCL
20115: 75 unit uPSI_DBXpressWeb;                   //Delphi VCL
20116: 76 unit uPSI_ShadowWnd;                     //Delphi VCL
20117: 77 unit uPSI_ToolWin;                       //Delphi VCL
20118: 78 unit uPSI_Tabs;                          //Delphi VCL
20119: 79 unit uPSI_JclGraphUtils;                 //JCL
20120: 80 unit uPSI_JclCounter;                    //JCL
20121: 81 unit uPSI_JclSysInfo;                    //JCL
20122: 82 unit uPSI_JclSecurity;                   //JCL
20123: 83 unit uPSI_JclFileUtils;                  //JCL
20124: 84 unit uPSI_IdUserAccounts;                //Indy
20125: 85 unit uPSI_IdAuthentication;              //Indy
20126: 86 unit uPSI_uTPLb_AES;                     //LockBox 3
20127: 87 unit uPSI_IdHashSHA1;                    //LockBox 3
20128: 88 unit uTPLb_BlockCipher;                  //LockBox 3
20129: 89 unit uPSI_ValEdit.pas;                   //Delphi VCL
20130: 90 unit uPSI_JvVCLUtils;                   //JCL
20131: 91 unit uPSI_JvDBUtil;                     //JCL
20132: 92 unit uPSI_JvDBUtils;                    //JCL
20133: 93 unit uPSI_JvAppUtils;                   //JCL
20134: 94 unit uPSI_JvCtrlUtils;                  //JCL
20135: 95 unit uPSI_JvFormToHtml;                 //JCL
20136: 96 unit uPSI_JvParsing;                    //JCL
20137: 97 unit uPSI_SerDlgs;                      //Toolbox
20138: 98 unit uPSI_Serial;                       //Toolbox
20139: 99 unit uPSI_JvComponent;                  //JCL
20140: 100 unit uPSI_JvCalc;                      //JCL
20141: 101 unit uPSI_JvBdeUtils;                  //JCL
20142: 102 unit uPSI_JvDateUtil;                  //JCL
20143: 103 unit uPSI_JvGenetic;                   //JCL
20144: 104 unit uPSI_JclBase;                     //JCL
20145: 105 unit uPSI_JvUtils;                     //JCL
20146: 106 unit uPSI_JvStrUtil;                   //JCL
20147: 107 unit uPSI_JvStrUtils;                  //JCL
20148: 108 unit uPSI_JvFileUtil;                  //JCL
20149: 109 unit uPSI_JvMemoryInfos;               //JCL
20150: 110 unit uPSI_JvComputerInfo;              //JCL
20151: 111 unit uPSI_JvgCommClasses;              //JCL
20152: 112 unit uPSI_JvgLogics;                   //JCL
20153: 113 unit uPSI_JvLED;                       //JCL
20154: 114 unit uPSI_JvTurtle;                    //JCL
20155: 115 unit uPSI_SortThds; unit uPSI_ThSort;  //maXbox
20156: 116 unit uPSI_JvgUtils;                    //JCL
20157: 117 unit uPSI_JvExprParser;                //JCL
20158: 118 unit uPSI_HexDump;                     //Borland
20159: 119 unit uPSI_DBLogDlg;                    //VCL
20160: 120 unit uPSI_SqlTimSt;                    //RTL
20161: 121 unit uPSI_JvHtmlParser;                //JCL
20162: 122 unit uPSI_JvgXMLSerializer;            //JCL
20163: 123 unit uPSI_JvJCLUtils;                  //JCL
20164: 124 unit uPSI_JvStrings;                   //JCL
20165: 125 unit uPSI_uTPLb_IntegerUtils;          //TurboPower
20166: 126 unit uPSI_uTPLb_HugeCardinal;          //TurboPower
20167: 127 unit uPSI_uTPLb_HugeCardinalUtils;     //TurboPower
20168: 128 unit uPSI_SynRegExpr;                  //SynEdit
20169: 129 unit uPSI_StUtils;                     //SysTools4
20170: 130 unit uPSI_StToHTML;                    //SysTools4
20171: 131 unit uPSI_StStrms;                     //SysTools4
20172: 132 unit uPSI_StFIN;                       //SysTools4
20173: 133 unit uPSI_StAstroP;                    //SysTools4
20174: 134 unit uPSI_StStat;                      //SysTools4
20175: 135 unit uPSI_StNetCon;                    //SysTools4
20176: 136 unit uPSI_StDecMth;                    //SysTools4
20177: 137 unit uPSI_StOStr;                      //SysTools4
20178: 138 unit uPSI_StPtrns;                     //SysTools4
20179: 139 unit uPSI_StNetMsg;                    //SysTools4
20180: 140 unit uPSI_StMath;                      //SysTools4
20181: 141 unit uPSI_StExpEng;                    //SysTools4
20182: 142 unit uPSI_StCRC;                       //SysTools4
20183: 143 unit uPSI_StExport,                    //SysTools4
20184: 144 unit uPSI_StExpLog,                    //SysTools4
20185: 145 unit uPSI_ActnList;                    //Delphi VCL
20186: 146 unit uPSI_jpeg;                        //Borland
20187: 147 unit uPSI_StRandom;                    //SysTools4
20188: 148 unit uPSI_StDict;                      //SysTools4
20189: 149 unit uPSI_StBCD;                       //SysTools4
20190: 150 unit uPSI_StTxtDat;                    //SysTools4
20191: 151 unit uPSI_StRegEx;                     //SysTools4
20192: 152 unit uPSI_IMouse;                      //VCL
20193: 153 unit uPSI_SyncObjs;                    //VCL
20194: 154 unit uPSI_AsyncCalls;                  //Hausladen
20195: 155 unit uPSI_ParallelJobs;                //Saraiva
20196: 156 unit uPSI_Variants;                    //VCL
20197: 157 unit uPSI_VarCmplx;                    //VCL Wolfram
20198: 158 unit uPSI_DTDSchema;                   //VCL
20199: 159 unit uPSI_ShLwApi;                     //Brakel
20200: 160 unit uPSI_IBUtils;                     //VCL
20201: 161 unit uPSI_CheckLst;                    //VCL
20202: 162 unit uPSI_JvSimpleXml;                 //JCL
```

```
20203: 163 unit uPSI_JclSimpleXml;                  //JCL
20204: 164 unit uPSI_JvXmlDatabase;                 //JCL
20205: 165 unit uPSI_JvMaxPixel;                    //JCL
20206: 166 unit uPSI_JvItemsSearchs;                //JCL
20207: 167 unit uPSI_StExpEng2;                     //SysTools4
20208: 168 unit uPSI_StGenLog;                      //SysTools4
20209: 169 unit uPSI_JvLogFile;                     //Jcl
20210: 170 unit uPSI_CPort;                         //ComPort Lib v4.11
20211: 171 unit uPSI_CPortCtl;                      //ComPort
20212: 172 unit uPSI_CPortEsc;                      //ComPort
20213: 173 unit BarCodeScaner;                      //ComPort
20214: 174 unit uPSI_JvGraph;                       //JCL
20215: 175 unit uPSI_JvComCtrls;                    //JCL
20216: 176 unit uPSI_GUITesting;                    //D Unit
20217: 177 unit uPSI_JvFindFiles;                   //JCL
20218: 178 unit uPSI_StSystem;                      //SysTools4
20219: 179 unit uPSI_JvKeyboardStates;              //JCL
20220: 180 unit uPSI_JvMail;                        //JCL
20221: 181 unit uPSI_JclConsole;                    //JCL
20222: 182 unit uPSI_JclLANMan;                     //JCL
20223: 183 unit uPSI_IdCustomHTTPServer;            //Indy
20224: 184 unit IdHTTPServer                        //Indy
20225: 185 unit uPSI_IdTCPServer;                   //Indy
20226: 186 unit uPSI_IdSocketHandle;                //Indy
20227: 187 unit uPSI_IdIOHandlerSocket;             //Indy
20228: 188 unit IdIOHandler;                        //Indy
20229: 189 unit uPSI_cutils;                        //Bloodshed
20230: 190 unit uPSI_BoldUtils;                     //boldsoft
20231: 191 unit uPSI_IdSimpleServer;                //Indy
20232: 192 unit uPSI_IdSSLOpenSSL;                  //Indy
20233: 193 unit uPSI_IdMultipartFormData;           //Indy
20234: 194 unit uPSI_SynURIOpener;                  //SynEdit
20235: 195 unit uPSI_PerlRegEx;                     //PCRE
20236: 196 unit uPSI_IdHeaderList;                  //Indy
20237: 197 unit uPSI_StFirst;                       //SysTools4
20238: 198 unit uPSI_JvCtrls;                       //JCL
20239: 199 unit uPSI_IdTrivialFTPBase;              //Indy
20240: 200 unit uPSI_IdTrivialFTP;                  //Indy
20241: 201 unit uPSI_IdUDPBase;                     //Indy
20242: 202 unit uPSI_IdUDPClient;                   //Indy
20243: 203 unit uPSI_utypes;                        //for DMath.DLL
20244: 204 unit uPSI_ShellAPI;                      //Borland
20245: 205 unit uPSI_IdRemoteCMDClient;             //Indy
20246: 206 unit uPSI_IdRemoteCMDServer;             //Indy
20247: 207 unit IdRexecServer;                      //Indy
20248: 208 unit IdRexec; (unit uPSI_IdRexec;)       //Indy
20249: 209 unit IdUDPServer;                        //Indy
20250: 210 unit IdTimeUDPServer;                    //Indy
20251: 211 unit IdTimeServer;                       //Indy
20252: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
20253: 213 unit uPSI_IdIPWatch;                     //Indy
20254: 214 unit uPSI_IdIrcServer;                   //Indy
20255: 215 unit uPSI_IdMessageCollection;           //Indy
20256: 216 unit uPSI_cPEM;                          //Fundamentals 4
20257: 217 unit uPSI_cFundamentUtils;               //Fundamentals 4
20258: 218 unit uPSI_uwinplot;                      //DMath
20259: 219 unit uPSI_xrtl_util_CPUUtils;           //ExtentedRTL
20260: 220 unit uPSI_GR32_System;                   //Graphics32
20261: 221 unit uPSI_cFileUtils;                    //Fundamentals 4
20262: 222 unit uPSI_cDateTime; (timemachine)      //Fundamentals 4
20263: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
20264: 224 unit uPSI_cRandom;                       //Fundamentals 4
20265: 225 unit uPSI_ueval;                         //DMath
20266: 226 unit uPSI_xrtl_net_URIUtils;             //ExtendedRTL
20267: 227 unit xrtl_net_URIUtils;                  //ExtendedRTL
20268: 228 unit uPSI_ufft;  (FFT)                   //DMath
20269: 229 unit uPSI_DBXChannel;                    //Delphi
20270: 230 unit uPSI_DBXIndyChannel;                //Delphi Indy
20271: 231 unit uPSI_xrtl_util_COMCat;              //ExtendedRTL
20272: 232 unit uPSI_xrtl_util_StrUtils;            //ExtendedRTL
20273: 233 unit uPSI_xrtl_util_VariantUtils;        //ExtendedRTL
20274: 234 unit uPSI_xrtl_util_FileUtils;           //ExtendedRTL
20275: 235 unit xrtl_util_Compat;                   //ExtendedRTL
20276: 236 unit uPSI_OleAuto;                       //Borland
20277: 237 unit uPSI_xrtl_util_COMUtils;            //ExtendedRTL
20278: 238 unit uPSI_CmAdmCtl;                      //Borland
20279: 239 unit uPSI_ValEdit2;                      //VCL
20280: 240 unit uPSI_GR32;  //Graphics32           //Graphics32
20281: 241 unit uPSI_GR32_Image;                    //Graphics32
20282: 242 unit uPSI_xrtl_util_TimeUtils;           //ExtendedRTL
20283: 243 unit uPSI_xrtl_util_TimeZone;            //ExtendedRTL
20284: 244 unit uPSI_xrtl_util_TimeStamp;           //ExtendedRTL
20285: 245 unit uPSI_xrtl_util_Map;                 //ExtendedRTL
20286: 246 unit uPSI_xrtl_util_Set;                 //ExtendedRTL
20287: 247 unit uPSI_CPortMonitor;                  //ComPort
20288: 248 unit uPSI_StIniStm;                      //SysTools4
20289: 249 unit uPSI_GR32_ExtImage;                 //Graphics32
20290: 250 unit uPSI_GR32_OrdinalMaps;              //Graphics32
20291: 251 unit uPSI_GR32_Rasterizers;              //Graphics32
```

```
20292: 252 unit uPSI_xrtl_util_Exception;          //ExtendedRTL
20293: 253 unit uPSI_xrtl_util_Value;              //ExtendedRTL
20294: 254 unit uPSI_xrtl_util_Compare;            //ExtendedRTL
20295: 255 unit uPSI_FlatSB;                       //VCL
20296: 256 unit uPSI_JvAnalogClock;                //JCL
20297: 257 unit uPSI_JvAlarms;                     //JCL
20298: 258 unit uPSI_JvSQLS;                       //JCL
20299: 259 unit uPSI_JvDBSecur;                    //JCL
20300: 260 unit uPSI_JvDBQBE;                      //JCL
20301: 261 unit uPSI_JvStarfield;                  //JCL
20302: 262 unit uPSI_JVCLMiscal;                   //JCL
20303: 263 unit uPSI_JvProfiler32;                 //JCL
20304: 264 unit uPSI_JvDirectories,                //JCL
20305: 265 unit uPSI_JclSchedule,                  //JCL
20306: 266 unit uPSI_JclSvcCtrl,                   //JCL
20307: 267 unit uPSI_JvSoundControl,               //JCL
20308: 268 unit uPSI_JvBDESQLScript,               //JCL
20309: 269 unit uPSI_JvgDigits,                    //JCL>
20310: 270 unit uPSI_ImgList;                      //TCustomImageList
20311: 271 unit uPSI_JclMIDI;                      //JCL>
20312: 272 unit uPSI_JclWinMidi;                   //JCL>
20313: 273 unit uPSI_JclNTFS;                      //JCL>
20314: 274 unit uPSI_JclAppInst;                   //JCL>
20315: 275 unit uPSI_JvRle;                        //JCL>
20316: 276 unit uPSI_JvRas32;                      //JCL>
20317: 277 unit uPSI_JvImageDrawThread,            //JCL>
20318: 278 unit uPSI_JvImageWindow,                //JCL>
20319: 279 unit uPSI_JvTransparentForm;            //JCL>
20320: 280 unit uPSI_JvWinDialogs;                 //JCL>
20321: 281 unit uPSI_JvSimLogic,                   //JCL>
20322: 282 unit uPSI_JvSimIndicator,               //JCL>
20323: 283 unit uPSI_JvSimPID,                     //JCL>
20324: 284 unit uPSI_JvSimPIDLinker,               //JCL>
20325: 285 unit uPSI_IdRFCReply;                   //Indy
20326: 286 unit uPSI_IdIdent;                      //Indy
20327: 287 unit uPSI_IdIdentServer;                //Indy
20328: 288 unit uPSI_JvPatchFile;                  //JCL
20329: 289 unit uPSI_StNetPfm;                     //SysTools4
20330: 290 unit uPSI_StNet;                        //SysTools4
20331: 291 unit uPSI_JclPeImage;                   //JCL
20332: 292 unit uPSI_JclPrint;                     //JCL
20333: 293 unit uPSI_JclMime;                      //JCL
20334: 294 unit uPSI_JvRichEdit;                   //JCL
20335: 295 unit uPSI_JvDBRichEd;                   //JCL
20336: 296 unit uPSI_JvDice;                       //JCL
20337: 297 unit uPSI_JvFloatEdit;                  //JCL 3.9.8
20338: 298 unit uPSI_JvDirFrm;                     //JCL
20339: 299 unit uPSI_JvDualList;                   //JCL
20340: 300 unit uPSI_JvSwitch;                     ////JCL
20341: 301 unit uPSI_JvTimerLst;                   ////JCL
20342: 302 unit uPSI_JvMemTable;                   //JCL
20343: 303 unit uPSI_JvObjStr;                     //JCL
20344: 304 unit uPSI_StLArr;                       //SysTools4
20345: 305 unit uPSI_StWmDCpy;                     //SysTools4
20346: 306 unit uPSI_StText;                       //SysTools4
20347: 307 unit uPSI_StNTLog;                      //SysTools4
20348: 308 unit uPSI_xrtl_math_Integer;            //ExtendedRTL
20349: 309 unit uPSI_JvImagPrvw;                   //JCL
20350: 310 unit uPSI_JvFormPatch;                  //JCL
20351: 311 unit uPSI_JvPicClip;                    //JCL
20352: 312 unit uPSI_JvDataConv;                   //JCL
20353: 313 unit uPSI_JvCpuUsage;                   //JCL
20354: 314 unit uPSI_JclUnitConv_mX2;              //JCL
20355: 315 unit JvDualListForm;                    //JCL
20356: 316 unit uPSI_JvCpuUsage2;                  //JCL
20357: 317 unit uPSI_JvParserForm;                 //JCL
20358: 318 unit uPSI_JvJanTreeView;                //JCL
20359: 319 unit uPSI_JvTransLED;                   //JCL
20360: 320 unit uPSI_JvPlaylist;                   //JCL
20361: 321 unit uPSI_JvFormAutoSize;               //JCL
20362: 322 unit uPSI_JvYearGridEditForm;           //JCL
20363: 323 unit uPSI_JvMarkupCommon;               //JCL
20364: 324 unit uPSI_JvChart;                      //JCL
20365: 325 unit uPSI_JvXPCore;                     //JCL
20366: 326 unit uPSI_JvXPCoreUtils;                //JCL
20367: 327 unit uPSI_StatsClasses;                 //mX4
20368: 328 unit uPSI_ExtCtrls2;                    //VCL
20369: 329 unit uPSI_JvUrlGrabbers;                //JCL
20370: 330 unit uPSI_JvXmlTree;                    //JCL
20371: 331 unit uPSI_JvWavePlayer;                 //JCL
20372: 332 unit uPSI_JvUnicodeCanvas;              //JCL
20373: 333 unit uPSI_JvTFUtils;                    //JCL
20374: 334 unit uPSI_IdServerIOHandler;            //Indy
20375: 335 unit uPSI_IdServerIOHandlerSocket;      //Indy
20376: 336 unit uPSI_IdMessageCoder;               //Indy
20377: 337 unit uPSI_IdMessageCoderMIME;           //Indy
20378: 338 unit uPSI_IdMIMETypes;                  //Indy
20379: 339 unit uPSI_JvConverter;                  //JCL
20380: 340 unit uPSI_JvCsvParse;                   //JCL
```

```
20381: 341 unit uPSI_umath;  unit uPSI_ugamma;        //DMath
20382: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
20383: 343 unit uPSI_JvDBGridExport;                  //JCL
20384: 344 unit uPSI_JvgExport;                       //JCL
20385: 345 unit uPSI_JvSerialMaker;                   //JCL
20386: 346 unit uPSI_JvWin32;                         //JCL
20387: 347 unit uPSI_JvPaintFX;                       //JCL
20388: 348 unit uPSI_JvOracleDataSet; (beta)          //JCL
20389: 349 unit uPSI_JvValidators; (preview)         //JCL
20390: 350 unit uPSI_JvNTEventLog;                    //JCL
20391: 351 unit uPSI_ShellZipTool;                    //mX4
20392: 352 unit uPSI_JvJoystick;                      //JCL
20393: 353 unit uPSI_JvMailSlots;                     //JCL
20394: 354 unit uPSI_JclComplex;                      //JCL
20395: 355 unit uPSI_SynPdf;                          //Synopse
20396: 356 unit uPSI_Registry;                        //VCL
20397: 357 unit uPSI_TlHelp32;                        //VCL
20398: 358 unit uPSI_JclRegistry;                     //JCL
20399: 359 unit uPSI_JvAirBrush;                      //JCL
20400: 360 unit uPSI_mORMotReport;                    //Synopse
20401: 361 unit uPSI_JclLocales;                      //JCL
20402: 362 unit uPSI_SynEdit;                         //SynEdit
20403: 363 unit uPSI_SynEditTypes;                    //SynEdit
20404: 364 unit uPSI_SynMacroRecorder;                //SynEdit
20405: 365 unit uPSI_LongIntList;                     //SynEdit
20406: 366 unit uPSI_devcutils;                       //DevC
20407: 367 unit uPSI_SynEditMiscClasses;              //SynEdit
20408: 368 unit uPSI_SynEditRegexSearch;              //SynEdit
20409: 369 unit uPSI_SynEditHighlighter;              //SynEdit
20410: 370 unit uPSI_SynHighlighterPas;               //SynEdit
20411: 371 unit uPSI_JvSearchFiles;                   //JCL
20412: 372 unit uPSI_SynHighlighterAny;               //Lazarus
20413: 373 unit uPSI_SynEditKeyCmds;                  //SynEdit
20414: 374 unit uPSI_SynEditMiscProcs,                //SynEdit
20415: 375 unit uPSI_SynEditKbdHandler                //SynEdit
20416: 376 unit uPSI_JvAppInst,                       //JCL
20417: 377 unit uPSI_JvAppEvent;                      //JCL
20418: 378 unit uPSI_JvAppCommand;                    //JCL
20419: 379 unit uPSI_JvAnimTitle;                     //JCL
20420: 380 unit uPSI_JvAnimatedImage;                 //JCL
20421: 381 unit uPSI_SynEditExport;                   //SynEdit
20422: 382 unit uPSI_SynExportHTML;                   //SynEdit
20423: 383 unit uPSI_SynExportRTF;                    //SynEdit
20424: 384 unit uPSI_SynEditSearch;                   //SynEdit
20425: 385 unit uPSI_fMain_back                       //maXbox;
20426: 386 unit uPSI_JvZoom;                          //JCL
20427: 387 unit uPSI_PMrand;                          //PM
20428: 388 unit uPSI_JvSticker;                       //JCL
20429: 389 unit uPSI_XmlVerySimple;                   //mX4
20430: 390 unit uPSI_Services;                        //ExtPascal
20431: 391 unit uPSI_ExtPascalUtils;                  //ExtPascal
20432: 392 unit uPSI_SocketsDelphi;                   //ExtPascal
20433: 393 unit uPSI_StBarC;                          //SysTools
20434: 394 unit uPSI_StDbBarC;                        //SysTools
20435: 395 unit uPSI_StBarPN;                         //SysTools
20436: 396 unit uPSI_StDbPNBC;                        //SysTools
20437: 397 unit uPSI_StDb2DBC;                        //SysTools
20438: 398 unit uPSI_StMoney;                         //SysTools
20439: 399 unit uPSI_JvForth;                         //JCL
20440: 400 unit uPSI_RestRequest;                     //mX4
20441: 401 unit uPSI_HttpRESTConnectionIndy;          //mX4
20442: 402 unit uPSI_JvXmlDatabase; //update         //JCL
20443: 403 unit uPSI_StAstro;                         //SysTools
20444: 404 unit uPSI_StSort;                          //SysTools
20445: 405 unit uPSI_StDate;                          //SysTools
20446: 406 unit uPSI_StDateSt;                        //SysTools
20447: 407 unit uPSI_StBase;                          //SysTools
20448: 408 unit uPSI_StVInfo;                         //SysTools
20449: 409 unit uPSI_JvBrowseFolder;                  //JCL
20450: 410 unit uPSI_JvBoxProcs;                      //JCL
20451: 411 unit uPSI_urandom; (unit uranuvag;)        //DMath
20452: 412 unit uPSI_usimann; (unit ugenalg;)         //DMath
20453: 413 unit uPSI_JvHighlighter;                   //JCL
20454: 414 unit uPSI_Diff;                            //mX4
20455: 415 unit uPSI_SpringWinAPI;                    //DSpring
20456: 416 unit uPSI_StBits;                          //SysTools
20457: 417 unit uPSI_TomDBQue;                        //mX4
20458: 418 unit uPSI_MultilangTranslator;             //mX4
20459: 419 unit uPSI_HyperLabel;                      //mX4
20460: 420 unit uPSI_Starter;                         //mX4
20461: 421 unit uPSI_FileAssocs;                      //devC
20462: 422 unit uPSI_devFileMonitorX;                 //devC
20463: 423 unit uPSI_devrun;                          //devC
20464: 424 unit uPSI_devExec;                         //devC
20465: 425 unit uPSI_oysUtils;                        //devC
20466: 426 unit uPSI_DosCommand;                      //devC
20467: 427 unit uPSI_CppTokenizer;                    //devC
20468: 428 unit uPSI_JvHLParser;                      //devC
20469: 429 unit uPSI_JclMapi;                         //JCL
```

```
20470: 430 unit uPSI_JclShell;                    //JCL
20471: 431 unit uPSI_JclCOM;                     //JCL
20472: 432 unit uPSI_GR32_Math;                  //Graphics32
20473: 433 unit uPSI_GR32_LowLevel;              //Graphics32
20474: 434 unit uPSI_SimpleHl;                   //mX4
20475: 435 unit uPSI_GR32_Filters,               //Graphics32
20476: 436 unit uPSI_GR32_VectorMaps;            //Graphics32
20477: 437 unit uPSI_cXMLFunctions;              //Fundamentals 4
20478: 438 unit uPSI_JvTimer;                    //JCL
20479: 439 unit uPSI_cHTTPUtils;                 //Fundamentals 4
20480: 440 unit uPSI_cTLSUtils;                  //Fundamentals 4
20481: 441 unit uPSI_JclGraphics;                //JCL
20482: 442 unit uPSI_JclSynch;                   //JCL
20483: 443 unit uPSI_IdTelnet;                   //Indy
20484: 444 unit uPSI_IdTelnetServer,             //Indy
20485: 445 unit uPSI_IdEcho,                     //Indy
20486: 446 unit uPSI_IdEchoServer,               //Indy
20487: 447 unit uPSI_IdEchoUDP,                  //Indy
20488: 448 unit uPSI_IdEchoUDPServer,           //Indy
20489: 449 unit uPSI_IdSocks,                    //Indy
20490: 450 unit uPSI_IdAntiFreezeBase;           //Indy
20491: 451 unit uPSI_IdHostnameServer;           //Indy
20492: 452 unit uPSI_IdTunnelCommon,             //Indy
20493: 453 unit uPSI_IdTunnelMaster,             //Indy
20494: 454 unit uPSI_IdTunnelSlave,              //Indy
20495: 455 unit uPSI_IdRSH,                      //Indy
20496: 456 unit uPSI_IdRSHServer,                //Indy
20497: 457 unit uPSI_Spring_Cryptography_Utils;  //Spring4Delphi
20498: 458 unit uPSI_MapReader,                  //devC
20499: 459 unit uPSI_LibTar,                     //devC
20500: 460 unit uPSI_IdStack;                    //Indy
20501: 461 unit uPSI_IdBlockCipherIntercept;     //Indy
20502: 462 unit uPSI_IdChargenServer;            //Indy
20503: 463 unit uPSI_IdFTPServer,                //Indy
20504: 464 unit uPSI_IdException,                //Indy
20505: 465 unit uPSI_utexplot;                   //DMath
20506: 466 unit uPSI_uwinstr;                    //DMath
20507: 467 unit uPSI_VarRecUtils;                //devC
20508: 468 unit uPSI_JvStringListToHtml,         //JCL
20509: 469 unit uPSI_JvStringHolder,             //JCL
20510: 470 unit uPSI_IdCoder;                    //Indy
20511: 471 unit uPSI_SynHighlighterDfm;          //Synedit
20512: 472 unit uHighlighterProcs; in 471        //Synedit
20513: 473 unit uPSI_LazFileUtils,               //LCL
20514: 474 unit uPSI_IDECmdLine;                 //LCL
20515: 475 unit uPSI_lazMasks;                   //LCL
20516: 476 unit uPSI_ip_misc;                    //mX4
20517: 477 unit uPSI_Barcode;                    //LCL
20518: 478 unit uPSI_SimpleXML;                  //LCL
20519: 479 unit uPSI_JclIniFiles;                //JCL
20520: 480 unit uPSI_D2XXUnit;  {$X-}            //FTDI
20521: 481 unit uPSI_JclDateTime;                //JCL
20522: 482 unit uPSI_JclEDI;                     //JCL
20523: 483 unit uPSI_JclMiscel2;                 //JCL
20524: 484 unit uPSI_JclValidation;              //JCL
20525: 485 unit uPSI_JclAnsiStrings; {-PString}  //JCL
20526: 486 unit uPSI_SynEditMiscProcs2;          //Synedit
20527: 487 unit uPSI_JclStreams;                 //JCL
20528: 488 unit uPSI_QRCode;                     //mX4
20529: 489 unit uPSI_BlockSocket;                //ExtPascal
20530: 490 unit uPSI_Masks,Utils                 //VCL
20531: 491 unit uPSI_synautil;                   //Synapse!
20532: 492 unit uPSI_JclMath_Class;              //JCL RTL
20533: 493 unit ugamdist; //Gamma function      //DMath
20534: 494 unit uibeta, ucorrel; //IBeta        //DMath
20535: 495 unit uPSI_SRMgr;                      //mX4
20536: 496 unit uPSI_HotLog;                     //mX4
20537: 497 unit uPSI_DebugBox;                   //mX4
20538: 498 unit uPSI_ustrings;                   //DMath
20539: 499 unit uPSI_uregtest;                   //DMath
20540: 500 unit uPSI_usimplex;                   //DMath
20541: 501 unit uPSI_uhyper;                     //DMath
20542: 502 unit uPSI_IdHL7;                      //Indy
20543: 503 unit uPSI_IdIPMCastBase,              //Indy
20544: 504 unit uPSI_IdIPMCastServer;            //Indy
20545: 505 unit uPSI_IdIPMCastClient;            //Indy
20546: 506 unit uPSI_unlfit; //nlregression     //DMath
20547: 507 unit uPSI_IdRawHeaders;               //Indy
20548: 508 unit uPSI_IdRawClient;                //Indy
20549: 509 unit uPSI_IdRawFunctions;             //Indy
20550: 510 unit uPSI_IdTCPStream;                //Indy
20551: 511 unit uPSI_IdSNPP;                     //Indy
20552: 512 unit uPSI_St2DBarC;                   //SysTools
20553: 513 unit uPSI_ImageWin;  //FTL           //VCL
20554: 514 unit uPSI_CustomDrawTreeView; //FTL   //VCL
20555: 515 unit uPSI_GraphWin;  //FTL           //VCL
20556: 516 unit uPSI_actionMain;  //FTL          //VCL
20557: 517 unit uPSI_StSpawn;                    //SysTools
20558: 518 unit uPSI_CtlPanel;                   //VCL
```

```
20559: 519 unit uPSI_IdLPR;                          //Indy
20560: 520 unit uPSI_SockRequestInterpreter;         //Indy
20561: 521 unit uPSI_ulambert;                      //DMath
20562: 522 unit uPSI_ucholesk;                      //DMath
20563: 523 unit uPSI_SimpleDS;                      //VCL
20564: 524 unit uPSI_DBXSqlScanner;                 //VCL
20565: 525 unit uPSI_DBXMetaDataUtil;              //VCL
20566: 526 unit uPSI_Chart;                        //TEE
20567: 527 unit uPSI_TeeProcs;                     //TEE
20568: 528 unit mXBDEUtils;                        //mX4
20569: 529 unit uPSI_MDIEdit;                      //VCL
20570: 530 unit uPSI_CopyPrsr;                     //VCL
20571: 531 unit uPSI_SockApp;                      //VCL
20572: 532 unit uPSI_AppEvnts;                     //VCL
20573: 533 unit uPSI_ExtActns;                     //VCL
20574: 534 unit uPSI_TeEngine;                     //TEE
20575: 535 unit uPSI_CoolMain; //browser           //VCL
20576: 536 unit uPSI_StCRC;                        //SysTools
20577: 537 unit uPSI_StDecMth2;                    //SysTools
20578: 538 unit uPSI_frmExportMain;                //Synedit
20579: 539 unit uPSI_SynDBEdit;                    //Synedit
20580: 540 unit uPSI_SynEditWildcardSearch;        //Synedit
20581: 541 unit uPSI_BoldComUtils;                 //BOLD
20582: 542 unit uPSI_BoldIsoDateTime;              //BOLD
20583: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils  //BOLD
20584: 544 unit uPSI_BoldXMLRequests;              //BOLD
20585: 545 unit uPSI_BoldStringList;               //BOLD
20586: 546 unit uPSI_BoldFileHandler;              //BOLD
20587: 547 unit uPSI_BoldContainers;               //BOLD
20588: 548 unit uPSI_BoldQueryUserDlg;             //BOLD
20589: 549 unit uPSI_BoldWinINet;                  //BOLD
20590: 550 unit uPSI_BoldQueue;                    //BOLD
20591: 551 unit uPSI_JvPcx;                        //JCL
20592: 552 unit uPSI_IdWhois;                      //Indy
20593: 553 unit uPSI_IdWhoIsServer;                //Indy
20594: 554 unit uPSI_IdGopher;                     //Indy
20595: 555 unit uPSI_IdDateTimeStamp;              //Indy
20596: 556 unit uPSI_IdDayTimeServer;              //Indy
20597: 557 unit uPSI_IdDayTimeUDP;                 //Indy
20598: 558 unit uPSI_IdDayTimeUDPServer;           //Indy
20599: 559 unit uPSI_IdDICTServer;                 //Indy
20600: 560 unit uPSI_IdDiscardServer;              //Indy
20601: 561 unit uPSI_IdDiscardUDPServer;           //Indy
20602: 562 unit uPSI_IdMappedFTP;                  //Indy
20603: 563 unit uPSI_IdMappedPortTCP;              //Indy
20604: 564 unit uPSI_IdGopherServer;               //Indy
20605: 565 unit uPSI_IdQotdServer;                 //Indy
20606: 566 unit uPSI_JvRgbToHtml;                  //JCL
20607: 567 unit uPSI_JvRemLog,                     //JCL
20608: 568 unit uPSI_JvSysComp;                    //JCL
20609: 569 unit uPSI_JvTMTL;                       //JCL
20610: 570 unit uPSI_JvWinampAPI;                  //JCL
20611: 571 unit uPSI_MSysUtils;                    //mX4
20612: 572 unit uPSI_ESBMaths;                     //ESB
20613: 573 unit uPSI_ESBMaths2;                    //ESB
20614: 574 unit uPSI_uLkJSON;                      //Lk
20615: 575 unit uPSI_ZURL;  //Zeos                 //Zeos
20616: 576 unit uPSI_ZSysUtils;                    //Zeos
20617: 577 unit unaUtils internals                 //UNA
20618: 578 unit uPSI_ZMatchPattern;                //Zeos
20619: 579 unit uPSI_ZClasses;                     //Zeos
20620: 580 unit uPSI_ZCollections;                 //Zeos
20621: 581 unit uPSI_ZEncoding;                    //Zeos
20622: 582 unit uPSI_IdRawBase;                    //Indy
20623: 583 unit uPSI_IdNTLM;                       //Indy
20624: 584 unit uPSI_IdNNTP;                       //Indy
20625: 585 unit uPSI_usniffer; //PortScanForm      //mX4
20626: 586 unit uPSI_IdCoderMIME;                  //Indy
20627: 587 unit uPSI_IdCoderUUE;                   //Indy
20628: 588 unit uPSI_IdCoderXXE;                   //Indy
20629: 589 unit uPSI_IdCoder3to4;                  //Indy
20630: 590 unit uPSI_IdCookie;                     //Indy
20631: 591 unit uPSI_IdCookieManager;              //Indy
20632: 592 unit uPSI_WDosSocketUtils;              //WDos
20633: 593 unit uPSI_WDosPlcUtils;                 //WDos
20634: 594 unit uPSI_WDosPorts;                    //WDos
20635: 595 unit uPSI_WDosResolvers;                //WDos
20636: 596 unit uPSI_WDosTimers;                   //WDos
20637: 597 unit uPSI_WDosPlcs;                     //WDos
20638: 598 unit uPSI_WDosPneumatics;               //WDos
20639: 599 unit uPSI_IdFingerServer;               //Indy
20640: 600 unit uPSI_IdDNSResolver;                //Indy
20641: 601 unit uPSI_IdHTTPWebBrokerBridge;        //Indy
20642: 602 unit uPSI_IdIntercept;                  //Indy
20643: 603 unit uPSI_IdIPMCastBase;                //Indy
20644: 604 unit uPSI_IdLogBase;                    //Indy
20645: 605 unit uPSI_IdIOHandlerStream;            //Indy
20646: 606 unit uPSI_IdMappedPortUDP;              //Indy
20647: 607 unit uPSI_IdQOTDUDPServer;              //Indy
```

```
20648: 608 unit uPSI_IdQOTDUDP;                    //Indy
20649: 609 unit uPSI_IdSysLog;                     //Indy
20650: 610 unit uPSI_IdSysLogServer;               //Indy
20651: 611 unit uPSI_IdSysLogMessage;              //Indy
20652: 612 unit uPSI_IdTimeServer;                 //Indy
20653: 613 unit uPSI_IdTimeUDP;                    //Indy
20654: 614 unit uPSI_IdTimeUDPServer;              //Indy
20655: 615 unit uPSI_IdUserAccounts;               //Indy
20656: 616 unit uPSI_TextUtils;                    //mX4
20657: 617 unit uPSI_MandelbrotEngine;             //mX4
20658: 618 unit uPSI_delphi_arduino_Unit1;         //mX4
20659: 619 unit uPSI_DTDSchema2;                   //mX4
20660: 620 unit uPSI_fplotMain;                    //DMath
20661: 621 unit uPSI_FindFileIter;                 //mX4
20662: 622 unit uPSI_PppState;  (JclStrHashMap)    //PPP
20663: 623 unit uPSI_PppParser;                    //PPP
20664: 624 unit uPSI_PppLexer;                     //PPP
20665: 625 unit uPSI_PCharUtils;                   //PPP
20666: 626 unit uPSI_uJSON;                        //WU
20667: 627 unit uPSI_JclStrHashMap;                //JCL
20668: 628 unit uPSI_JclHookExcept;                //JCL
20669: 629 unit uPSI_EncdDecd;                     //VCL
20670: 630 unit uPSI_SockAppReg;                   //VCL
20671: 631 unit uPSI_PJFileHandle;                 //PJ
20672: 632 unit uPSI_PJEnvVars;                    //PJ
20673: 633 unit uPSI_PJPipe;                       //PJ
20674: 634 unit uPSI_PJPipeFilters;                //PJ
20675: 635 unit uPSI_PJConsoleApp;                 //PJ
20676: 636 unit uPSI_UConsoleAppEx;                //PJ
20677: 637 unit uPSI_DbxSocketChannelNative,       //VCL
20678: 638 unit uPSI_DbxDataGenerator,             //VCL
20679: 639 unit uPSI_DBXClient;                    //VCL
20680: 640 unit uPSI_IdLogEvent;                   //Indy
20681: 641 unit uPSI_Reversi;                      //mX4
20682: 642 unit uPSI_Geometry;                     //mX4
20683: 643 unit uPSI_IdSMTPServer;                 //Indy
20684: 644 unit uPSI_Textures;                     //mX4
20685: 645 unit uPSI_IBX;                          //VCL
20686: 646 unit uPSI_IWDBCommon;                   //VCL
20687: 647 unit uPSI_SortGrid;                     //mX4
20688: 648 unit uPSI_IB;                           //VCL
20689: 649 unit uPSI_IBScript;                     //VCL
20690: 650 unit uPSI_JvCSVBaseControls;            //JCL
20691: 651 unit uPSI_Jvg3DColors;                  //JCL
20692: 652 unit uPSI_JvHLEditor;  //beat           //JCL
20693: 653 unit uPSI_JvShellHook;                  //JCL
20694: 654 unit uPSI_DBCommon2                     //VCL
20695: 655 unit uPSI_JvSHFileOperation;            //JCL
20696: 656 unit uPSI_uFilexport;                   //mX4
20697: 657 unit uPSI_JvDialogs;                    //JCL
20698: 658 unit uPSI_JvDBTreeView;                 //JCL
20699: 659 unit uPSI_JvDBUltimGrid;                //JCL
20700: 660 unit uPSI_JvDBQueryParamsForm;          //JCL
20701: 661 unit uPSI_JvExControls;                 //JCL
20702: 662 unit uPSI_JvBDEMemTable;                //JCL
20703: 663 unit uPSI_JvCommStatus;                 //JCL
20704: 664 unit uPSI_JvMailSlots2;                 //JCL
20705: 665 unit uPSI_JvgWinMask;                   //JCL
20706: 666 unit uPSI_StEclpse;                     //SysTools
20707: 667 unit uPSI_StMime;                       //SysTools
20708: 668 unit uPSI_StList;                       //SysTools
20709: 669 unit uPSI_StMerge;                      //SysTools
20710: 670 unit uPSI_StStrS;                       //SysTools
20711: 671 unit uPSI_StTree,                       //SysTools
20712: 672 unit uPSI_StVArr;                       //SysTools
20713: 673 unit uPSI_StRegIni;                     //SysTools
20714: 674 unit uPSI_urkf;                         //DMath
20715: 675 unit uPSI_usvd;                         //DMath
20716: 676 unit uPSI_DepWalkUtils;                 //JCL
20717: 677 unit uPSI_OptionsFrm;                   //JCL
20718: 678 unit yuvconverts;                       //mX4
20719: 679 uPSI_JvPropAutoSave;                    //JCL
20720: 680 uPSI_AclAPI;                            //alcinoe
20721: 681 uPSI_AviCap;                            //alcinoe
20722: 682 uPSI_ALAVLBinaryTree;                   //alcinoe
20723: 683 uPSI_ALFcnMisc;                         //alcinoe
20724: 684 uPSI_ALStringList;                      //alcinoe
20725: 685 uPSI_ALQuickSortList;                   //alcinoe
20726: 686 uPSI_ALStaticText;                      //alcinoe
20727: 687 uPSI_ALJSONDoc;                         //alcinoe
20728: 688 uPSI_ALGSMComm;                         //alcinoe
20729: 689 uPSI_ALWindows;                         //alcinoe
20730: 690 uPSI_ALMultiPartFormDataParser;         //alcinoe
20731: 691 uPSI_ALHttpCommon;                      //alcinoe
20732: 692 uPSI_ALWebSpider,                       //alcinoe
20733: 693 uPSI_ALHttpClient;                      //alcinoe
20734: 694 uPSI_ALFcnHTML;                         //alcinoe
20735: 695 uPSI_ALFTPClient;                       //alcinoe
20736: 696 uPSI_ALInternetMessageCommon;           //alcinoe
```

```
20737: 697 uPSI_ALWininetHttpClient;              //alcinoe
20738: 698 uPSI_ALWinInetFTPClient;               //alcinoe
20739: 699 uPSI_ALWinHttpWrapper;                 //alcinoe
20740: 700 uPSI_ALWinHttpClient;                  //alcinoe
20741: 701 uPSI_ALFcnWinSock;                     //alcinoe
20742: 702 uPSI_ALFcnSQL;                         //alcinoe
20743: 703 uPSI_ALFcnCGI;                         //alcinoe
20744: 704 uPSI_ALFcnExecute;                     //alcinoe
20745: 705 uPSI_ALFcnFile;                        //alcinoe
20746: 706 uPSI_ALFcnMime;                        //alcinoe
20747: 707 uPSI_ALPhpRunner;                      //alcinoe
20748: 708 uPSI_ALGraphic;                        //alcinoe
20749: 709 uPSI_ALIniFiles;                       //alcinoe
20750: 710 uPSI_ALMemCachedClient;                //alcinoe
20751: 711 unit uPSI_MyGrids;                     //mX4
20752: 712 uPSI_ALMultiPartMixedParser            //alcinoe
20753: 713 uPSI_ALSMTPClient                      //alcinoe
20754: 714 uPSI_ALNNTPClient;                     //alcinoe
20755: 715 uPSI_ALHintBalloon;                    //alcinoe
20756: 716 unit uPSI_ALXmlDoc;                    //alcinoe
20757: 717 unit uPSI_IPCThrd;                     //VCL
20758: 718 unit uPSI_MonForm;                     //VCL
20759: 719 unit uPSI_TeCanvas;                    //Orpheus
20760: 720 unit uPSI_Ovcmisc;                     //Orpheus
20761: 721 unit uPSI_ovcfiler;                    //Orpheus
20762: 722 unit uPSI_ovcstate;                    //Orpheus
20763: 723 unit uPSI_ovccoco;                     //Orpheus
20764: 724 unit uPSI_ovcrvexp;                    //Orpheus
20765: 725 unit uPSI_OvcFormatSettings;           //Orpheus
20766: 726 unit uPSI_OvcUtils;                    //Orpheus
20767: 727 unit uPSI_ovcstore;                    //Orpheus
20768: 728 unit uPSI_ovcstr;                      //Orpheus
20769: 729 unit uPSI_ovcmru;                      //Orpheus
20770: 730 unit uPSI_ovccmd;                      //Orpheus
20771: 731 unit uPSI_ovctimer;                    //Orpheus
20772: 732 unit uPSI_ovcintl;                     //Orpheus
20773: 733 uPSI_AfCircularBuffer;                 //AsyncFree
20774: 734 uPSI_AfUtils;                          //AsyncFree
20775: 735 uPSI_AfSafeSync;                       //AsyncFree
20776: 736 uPSI_AfComPortCore;                    //AsyncFree
20777: 737 uPSI_AfComPort;                        //AsyncFree
20778: 738 uPSI_AfPortControls;                   //AsyncFree
20779: 739 uPSI_AfDataDispatcher;                 //AsyncFree
20780: 740 uPSI_AfViewers;                        //AsyncFree
20781: 741 uPSI_AfDataTerminal;                   //AsyncFree
20782: 742 uPSI_SimplePortMain;                   //AsyncFree
20783: 743 unit uPSI_ovcclock;                    //Orpheus
20784: 744 unit uPSI_o32intlst;                   //Orpheus
20785: 745 unit uPSI_o32ledlabel;                 //Orpheus
20786: 746 unit uPSI_AlMySqlClient;               //alcinoe
20787: 747 unit uPSI_ALFBXClient;                 //alcinoe
20788: 748 unit uPSI_ALFcnSQL;                    //alcinoe
20789: 749 unit uPSI_AsyncTimer;                  //mX4
20790: 750 unit uPSI_ApplicationFileIO;           //mX4
20791: 751 unit uPSI_PsAPI;                       //VCLé
20792: 752 uPSI_ovcuser;                          //Orpheus
20793: 753 uPSI_ovcurl;                           //Orpheus
20794: 754 uPSI_ovcvlb;                           //Orpheus
20795: 755 uPSI_ovccolor;                         //Orpheus
20796: 756 uPSI_ALFBXLib,                         //alcinoe
20797: 757 uPSI_ovcmeter;                         //Orpheus
20798: 758 uPSI_ovcpeakm;                         //Orpheus
20799: 759 uPSI_O32BGSty;                         //Orpheus
20800: 760 uPSI_ovcBidi;                          //Orpheus
20801: 761 uPSI_ovctcary;                         //Orpheus
20802: 762 uPSI_DXPUtils;                         //mX4
20803: 763 uPSI_ALMultiPartBaseParser;            //alcinoe
20804: 764 uPSI_ALMultiPartAlternativeParser;     //alcinoe
20805: 765 uPSI_ALPOP3Client;                     //alcinoe
20806: 766 uPSI_SmallUtils;                       //mX4
20807: 767 uPSI_MakeApp;                          //mX4
20808: 768 uPSI_O32MouseMon;                      //Orpheus
20809: 769 uPSI_OvcCache;                         //Orpheus
20810: 770 uPSI_ovccalc;                          //Orpheus
20811: 771 uPSI_Joystick,                         //OpenGL
20812: 772 uPSI_ScreenSaver;                      //OpenGL
20813: 773 uPSI_XCollection,                      //OpenGL
20814: 774 uPSI_Polynomials,                      //OpenGL
20815: 775 uPSI_PersistentClasses, //9.86        //OpenGL
20816: 776 uPSI_VectorLists;                      //OpenGL
20817: 777 uPSI_XOpenGL,                          //OpenGL
20818: 778 uPSI_MeshUtils;                        //OpenGL
20819: 779 unit uPSI_JclSysUtils;                 //JCL
20820: 780 unit uPSI_JclBorlandTools;             //JCL
20821: 781 unit JclFileUtils_max;                 //JCL
20822: 782 uPSI_AfDataControls,                   //AsyncFree
20823: 783 uPSI_GLSilhouette;                     //OpenGL
20824: 784 uPSI_JclSysUtils_class;                //JCL
20825: 785 uPSI_JclFileUtils_class;               //JCL
```

```
20826: 786 uPSI_FileUtil;                    //JCL
20827: 787 uPSI_changefind;                  //mX4
20828: 788 uPSI_cmdIntf;                     //mX4
20829: 789 uPSI_fservice;                    //mX4
20830: 790 uPSI_Keyboard;                    //OpenGL
20831: 791 uPSI_VRMLParser,                  //OpenGL
20832: 792 uPSI_GLFileVRML,                  //OpenGL
20833: 793 uPSI_Octree;                      //OpenGL
20834: 794 uPSI_GLPolyhedron,               //OpenGL
20835: 795 uPSI_GLCrossPlatform;            //OpenGL
20836: 796 uPSI_GLParticles;                //OpenGL
20837: 797 uPSI_GLNavigator;                //OpenGL
20838: 798 uPSI_GLStarRecord;               //OpenGL
20839: 799 uPSI_GLTextureCombiners;         //OpenGL
20840: 800 uPSI_GLCanvas;                   //OpenGL
20841: 801 uPSI_GeometryBB;                 //OpenGL
20842: 802 uPSI_GeometryCoordinates;        //OpenGL
20843: 803 uPSI_VectorGeometry;             //OpenGL
20844: 804 uPSI_BumpMapping;                //OpenGL
20845: 805 uPSI_TGA;                        //OpenGL
20846: 806 uPSI_GLVectorFileObjects;        //OpenGL
20847: 807 uPSI_IMM;                        //VCL
20848: 808 uPSI_CategoryButtons;            //VCL
20849: 809 uPSI_ButtonGroup;                //VCL
20850: 810 uPSI_DbExcept;                   //VCL
20851: 811 uPSI_AxCtrls;                    //VCL
20852: 812 uPSI_GL_actorUnit1;              //OpenGL
20853: 813 uPSI_StdVCL;                     //VCL
20854: 814 unit CurvesAndSurfaces;          //OpenGL
20855: 815 uPSI_DataAwareMain;              //AsyncFree
20856: 816 uPSI_TabNotBk;                   //VCL
20857: 817 uPSI_udwsfiler;                  //mX4
20858: 818 uPSI_synaip;                     //Synapse!
20859: 819 uPSI_synacode;                   //Synapse
20860: 820 uPSI_synachar;                   //Synapse
20861: 821 uPSI_synamisc;                   //Synapse
20862: 822 uPSI_synaser;                    //Synapse
20863: 823 uPSI_synaicnv;                   //Synapse
20864: 824 uPSI_tlntsend;                   //Synapse
20865: 825 uPSI_pingsend;                   //Synapse
20866: 826 uPSI_blcksock;                   //Synapse
20867: 827 uPSI_asn1util;                   //Synapse
20868: 828 uPSI_dnssend;                    //Synapse
20869: 829 uPSI_clamsend;                   //Synapse
20870: 830 uPSI_ldapsend;                   //Synapse
20871: 831 uPSI_mimemess;                   //Synapse
20872: 832 uPSI_slogsend;                   //Synapse
20873: 833 uPSI_mimepart;                   //Synapse
20874: 834 uPSI_mimeinln;                   //Synapse
20875: 835 uPSI_ftpsend,                    //Synapse
20876: 836 uPSI_ftptsend;                   //Synapse
20877: 837 uPSI_httpsend;                   //Synapse
20878: 838 uPSI_sntpsend;                   //Synapse
20879: 839 uPSI_smtpsend;                   //Synapse
20880: 840 uPSI_snmpsend;                   //Synapse
20881: 841 uPSI_imapsend;                   //Synapse
20882: 842 uPSI_pop3send;                   //Synapse
20883: 843 uPSI_nntpsend;                   //Synapse
20884: 844 uPSI_ssl_cryptlib;               //Synapse
20885: 845 uPSI_ssl_openssl;                //Synapse
20886: 846 uPSI_synhttp_daemon;             //Synapse
20887: 847 uPSI_NetWork;                    //mX4
20888: 848 uPSI_PingThread;                 //Synapse
20889: 849 uPSI_JvThreadTimer;              //JCL
20890: 850 unit uPSI_wwSystem;              //InfoPower
20891: 851 unit uPSI_IdComponent;           //Indy
20892: 852 unit uPSI_IdIOHandlerThrottle;   //Indy
20893: 853 unit uPSI_Themes;                //VCL
20894: 854 unit uPSI_StdStyleActnCtrls;     //VCL
20895: 855 unit uPSI_UDDIHelper;            //VCL
20896: 856 unit uPSI_IdIMAP4Server;         //Indy
20897: 857 uPSI_VariantSymbolTable,         //VCL //3.9.9.92
20898: 858 uPSI_udf_glob,                   //mX4
20899: 859 uPSI_TabGrid,                    //VCL
20900: 860 uPSI_JsDBTreeView,               //mX4
20901: 861 uPSI_JsSendMail,                 //mX4
20902: 862 uPSI_dbTvRecordList,             //mX4
20903: 863 uPSI_TreeVwEx,                   //mX4
20904: 864 uPSI_ECDataLink,                 //mX4
20905: 865 uPSI_dbTree,                     //mX4
20906: 866 uPSI_dbTreeCBox,                 //mX4
20907: 867 unit uPSI_Debug; //TfrmDebug     //mX4
20908: 868 uPSI_TimeFncs;                   //mX4
20909: 869 uPSI_FileIntf,                   //VCL
20910: 870 uPSI_SockTransport,              //RTL
20911: 871 unit uPSI_WinInet;               //RTL
20912: 872 unit uPSI_Wwstr;                 //mX4
20913: 873 uPSI_DBLookup,                   //VCL
20914: 874 uPSI_Hotspot,                    //mX4
```

```
20915: 875 uPSI_HList; //History List                //mX4
20916: 876 unit uPSI_DrTable;                        //VCL
20917: 877 uPSI_TConnect,                           //VCL
20918: 978 uPSI_DataBkr,                            //VCL
20919: 979 uPSI_HTTPIntr;                           //VCL
20920: 980 unit uPSI_Mathbox;                       //mX4
20921: 881 uPSI_cyIndy,                            //cY
20922: 882 uPSI_cySysUtils,                        //cY
20923: 883 uPSI_cyWinUtils,                        //cY
20924: 884 uPSI_cyStrUtils,                        //cY
20925: 885 uPSI_cyObjUtils,                        //cY
20926: 886 uPSI_cyDateUtils,                       //cY
20927: 887 uPSI_cyBDE,                             //cY
20928: 888 uPSI_cyClasses,                         //cY
20929: 889 uPSI_cyGraphics,  //3.9.9.94_2          //cY
20930: 890 unit uPSI_cyTypes;                       //cY
20931: 891 uPSI_JvDateTimePicker,                  //JCL
20932: 892 uPSI_JvCreateProcess,                   //JCL
20933: 893 uPSI_JvEasterEgg,                       //JCL
20934: 894 uPSI_WinSvc,  //3.9.9.94_3              //VCL
20935: 895 uPSI_SvcMgr                            //VCL
20936: 896 unit uPSI_JvPickDate;                   //JCL
20937: 897 unit uPSI_JvNotify;                     //JCL
20938: 898 uPSI_JvStrHlder                         //JCL
20939: 899 unit uPSI_JclNTFS2;                     //JCL
20940: 900 uPSI_Jcl8087 //math coprocessor        //JCL
20941: 901 uPSI_JvAddPrinter                       //JCL
20942: 902 uPSI_JvCabFile                          //JCL
20943: 903 uPSI_JvDataEmbedded;                    //JCL
20944: 904 unit uPSI_U_HexView;                    //mX4
20945: 905 uPSI_UWavein4,                          //mX4
20946: 906 uPSI_AMixer,                            //mX4
20947: 907 uPSI_JvaScrollText,                     //mX4
20948: 908 uPSI_JvArrow,                           //mX4
20949: 909 unit uPSI_UrlMon;                       //mX4
20950: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'  //mX4
20951: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;  //DFF
20952: 912 unit uPSI_DFFUtils;                     //DFF
20953: 913 unit uPSI_MathsLib;                     //DFF
20954: 914 uPSI_UIntList;                          //DFF
20955: 915 uPSI_UGetParens;                        //DFF
20956: 916 unit uPSI_UGeometry;                    //DFF
20957: 917 unit uPSI_UAstronomy;                   //DFF
20958: 918 unit uPSI_UCardComponentV2;             //DFF
20959: 919 unit uPSI_UTGraphSearch;                //DFF
20960: 920 unit uPSI_UParser10;                    //DFF
20961: 921 unit uPSI_cyIEUtils;                    //cY
20962: 922 unit uPSI_UcomboV2;                     //DFF
20963: 923 uPSI_cyBaseComm,                        //cY
20964: 924 uPSI_cyAppInstances,                    //cY
20965: 925 uPSI_cyAttract,                         //cY
20966: 926 uPSI_cyDERUtils                         //cY
20967: 927 unit uPSI_cyDocER;                      //cY
20968: 928 unit uPSI_ODBC;                         //mX
20969: 929 unit uPSI_AssocExec;                    //mX
20970: 930 uPSI_cyBaseCommRoomConnector,           //cY
20971: 931 uPSI_cyCommRoomConnector,               //cY
20972: 932 uPSI_cyCommunicate,                     //cY
20973: 933 uPSI_cyImage;                           //cY
20974: 934 uPSI_cyBaseContainer                    //cY
20975: 935 uPSI_cyModalContainer,                  //cY
20976: 936 uPSI_cyFlyingContainer;                 //cY
20977: 937 uPSI_RegStr,                            //VCL
20978: 938 uPSI_HtmlHelpViewer;                    //VCL
20979: 939 unit uPSI_cyIniForm                     //cY
20980: 940 unit uPSI_cyVirtualGrid;                //cY
20981:
20982:
20983:
20984:
20985: ////////////////////////////////////////////////////////////////////////////
20986: //Form Template Library FTL
20987: ////////////////////////////////////////////////////////////////////////////
20988:
20989: 30 FTL For Form Building out of the Script, eg. 399_form_templates.txt
20990:
20991: 045 unit uPSI_VListView                      TFormListView;
20992: 263 unit uPSI_JvProfiler32;                  TProfReport
20993: 270 unit uPSI_ImgList;                       TCustomImageList
20994: 278 unit uPSI_JvImageWindow;                 TJvImageWindow
20995: 317 unit uPSI_JvParserForm;                  TJvHTMLParserForm
20996: 497 unit uPSI_DebugBox;                      TDebugBox
20997: 513 unit uPSI_ImageWin;                      TImageForm, TImageForm2
20998: 514 unit uPSI_CustomDrawTreeView;           TCustomDrawForm
20999: 515 unit uPSI_GraphWin;                     TGraphWinForm
21000: 516 unit uPSI_actionMain;                   TActionForm
21001: 518 unit uPSI_CtlPanel;                     TAppletApplication
21002: 529 unit uPSI_MDIEdit;                      TEditForm
21003: 535 unit uPSI_CoolMain; {browser}           TWebMainForm
```

```
21004: 538 unit uPSI_frmExportMain;                    TSynexportForm
21005: 585 unit uPSI_usniffer; {//PortScanForm}        TSniffForm
21006: 600 unit uPSI_ThreadForm;                       TThreadSortForm;
21007: 618 unit uPSI_delphi_arduino_Unit1;             TLEDForm
21008: 620 unit uPSI_fplotMain;                        TfplotForm1
21009: 660 unit uPSI_JvDBQueryParamsForm;              TJvQueryParamsDialog
21010: 677 unit uPSI_OptionsFrm;                       TfrmOptions;
21011: 718 unit uPSI_MonForm;                         TMonitorForm
21012: 742 unit uPSI_SimplePortMain;                   TPortForm1
21013: 770 unit uPSI_ovccalc;                          TOvcCalculator  //widget
21014: 810 unit uPSI_DbExcept;                         TDbEngineErrorDlg
21015: 812 unit uPSI_GL_actorUnit1;                    TglActorForm1  //OpenGL Robot
21016: 846 unit uPSI_synhttp_daemon;                   TTCPHttpDaemon, TTCPHttpThrd, TPingThread
21017: 867 unit uPSI_Debug;                            TfrmDebug
21018: 904 unit uPSI_U_HexView;                        THexForm2
21019: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)     TOscfrmMain
21020:
21021:
21022: ex.:with TEditForm.create(self) do begin
21023:       caption:= 'Template Form Tester';
21024:       FormStyle:= fsStayOnTop;
21025:       with editor do begin
21026:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
21027:         SelStart:= 0;
21028:         Modified:= False;
21029:       end;
21030:     end;
21031:   with TWebMainForm.create(self) do begin
21032:     URLs.Text:= 'http://www.kleiner.ch';
21033:     URLsClick(self); Show;
21034:   end;
21035:   with TSynexportForm.create(self) do begin
21036:     Caption:= 'Synexport HTML RTF tester';
21037:     Show;
21038:   end;
21039:   with TThreadSortForm.create(self) do begin
21040:      showmodal; free;
21041:   end;
21042:
21043:   with TCustomDrawForm.create(self) do begin
21044:        width:=820; height:=820;
21045:        image1.height:= 600; //add properties
21046:        image1.picture.bitmap:= image2.picture.bitmap;
21047:        //SelectionBackground1Click(self) CustomDraw1Click(self);
21048:        Background1.click;
21049:        bitmap1.click;
21050:        Tile1.click;
21051:        Showmodal;
21052:        Free;
21053:      end;
21054:
21055:    with TfplotForm1.Create(self) do begin
21056:      BtnPlotClick(self);
21057:      Showmodal; Free;
21058:    end;
21059:
21060:   with TOvcCalculator.create(self) do begin
21061:     parent:= aForm;
21062:    //free;
21063:     setbounds(550,510,200,150);
21064:     displaystr:= 'maXcalc';
21065:   end;
21066:
21067:   with THexForm2.Create(self) do begin
21068:      ShowModal;
21069:      Free;
21070:   end;
21071:
21072: function CheckBox: string;
21073: var idHTTP: TIDHTTP;
21074:  begin
21075:    result:= 'version not found';
21076:    if IsInternet then begin
21077:      idHTTP:= TIdHTTP.Create(NIL);
21078:      try
21079:        result:= idHTTP.Get(MXVERSIONFILE2);
21080:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
21081:        if result = MBVER2 then begin
21082:          //output.Font.Style:= [fsbold];
21083:          //Speak(' A new Version '+vstr+' of max box is available! ');
21084:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
21085:        end;
21086:       //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
21087:      finally
21088:        idHTTP.Free
21089:    end;
21090:  end;
21091: end;
21092:
```

```
21093:
21094:
21095:
21096:
21097: ////////////////////////////////////////////////////////////////////////////
21098: All maXbox Tutorials Table of Content 2014
21099: ////////////////////////////////////////////////////////////////////////////
21100:     Tutorial 00 Function-Coding  (Blix the Programmer)
21101:     Tutorial 01 Procedural-Coding
21102:     Tutorial 02 OO-Programming
21103:     Tutorial 03 Modular Coding
21104:     Tutorial 04 UML Use Case Coding
21105:     Tutorial 05 Internet Coding
21106:     Tutorial 06 Network Coding
21107:     Tutorial 07 Game Graphics Coding
21108:     Tutorial 08 Operating System Coding
21109:     Tutorial 09 Database Coding
21110:     Tutorial 10 Statistic Coding
21111:     Tutorial 11 Forms Coding
21112:     Tutorial 12 SQL DB Coding
21113:     Tutorial 13 Crypto Coding
21114:     Tutorial 14 Parallel Coding
21115:     Tutorial 15 Serial RS232 Coding
21116:     Tutorial 16 Event Driven Coding
21117:     Tutorial 17 Web Server Coding
21118:     Tutorial 18 Arduino System Coding
21119:     Tutorial 18_3 RGB LED System Coding
21120:     Tutorial 19 WinCOM /Arduino Coding
21121:     Tutorial 20 Regular Expressions RegEx
21122:     Tutorial 21 Android Coding (coming 2013)
21123:     Tutorial 22 Services Programming
21124:     Tutorial 23 Real Time Systems
21125:     Tutorial 24 Clean Code
21126:     Tutorial 25 maXbox Configuration I+II
21127:     Tutorial 26 Socket Programming with TCP
21128:     Tutorial 27 XML & TreeView
21129:     Tutorial 28 DLL Coding (available)
21130:     Tutorial 29 UML Scripting (2014)
21131:     Tutorial 30 Web of Things (2014)
21132:     Tutorial 31 Closures (coming 2014)
21133:     Tutorial 32 SQL Firebird (coming 2014)
21134:     Tutorial 33 Oscilloscope (coming 2015)
21135:     Tutorial 34 GPS Navigation (coming 2015)
21136:
21137:
21138: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
21139: using Docu for this file is maxbox_functions_all.pdf
21140: PEP - Pascal Education Program  Lib Lab
21141:
21142:
21143: http://stackoverflow.com/tags/pascalscript/hot
21144: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
21145: http://sourceforge.net/projects/maXbox   #locs:51620
21146: http://sourceforge.net/apps/mediawiki/maXbox
21147: http://www.blaisepascal.eu/
21148: https://github.com/maxkleiner/maXbox3.git
21149: http://www.heise.de/download/maxbox-1176464.html
21150: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
21151: https://www.facebook.com/pages/Programming-maXbox/166844836691703
21152: http://www.softwareschule.ch/arduino_training.pdf
21153:
21154:
21155:   ---- bigbitbox code_cleared_checked----
```