```
 1: ********************************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9.98 R8
 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: 22985728 V3.9.9.98 October 2014 To EKON/BASTA/JAX/IBZ/SWS/EU/DT
10: ****************Now the Funclist*****************************************
11: Funclist Function : 13419 //10766 //10165 (7648)
12: ****************Now the Proclist*****************************************
13: Proclist Procedure Size is: 8289  //6792 //6401 4752
14: ****************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1337 //995 //
16: def head:max: maXbox7: 10.09.2014 09:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: --------------------------------------------------------------------------------
20: Funclist total Size all is: 23045! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 23364
22: ASize of EXE: 22985728 (22041600) (16586240) (13511680)
23: SHA1 Hash of maXbox 3.9.9.98: 7C600FFC801FFF2AAEC83DD220202DDF5271C64D
24:
25: --------------------------------------------------------------------------------
26: --------------------------------------------------------------------------------
27: ////////////////////////////////////////////////////////////////////////////////
28:
29:
30: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
31: Function *************Now the Funclist*****************
32: function  GetResStringChecked(Ident: string; const Args: array of const): string
33: Function ( Index : Longint) : Integer
34: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
35: Function _CheckAutoResult( ResultCode : HResult) : HResult
36: function _T(Name: tbtString): Variant;
37: function ABNFToText(const AText : String) : String
38: Function Abs(e : Extended) : Extended;
39: Function Ackermann( const A, B : Integer) : Integer
40: Function AcquireLayoutLock : Boolean
41: Function ActionByName( const AName : string) : TWebActionItem
42: Function ACTIVEBUFFER : PCHAR
43: Function Add : TAggregate
44: function Add : TCollectionItem
45: Function Add : TColumn
46: Function Add : TComboExItem
47: Function Add : TCookie
48: Function Add : TCoolBand
49: Function Add : TFavoriteLinkItem
50: Function Add : TFileTypeItem
51: Function Add : THeaderSection
52: Function Add : THTMLTableColumn
53: Function Add : TIdEMailAddressItem
54: Function Add : TIdMessagePart
55: Function Add : TIdUserAccount
56: Function Add : TListColumn
57: Function Add : TListItem
58: Function Add : TStatusPanel
59: Function Add : TTaskDialogBaseButtonItem
60: Function Add : TWebActionItem
61: Function Add : TWorkArea
62: Function Add( AClass : TClass) : Integer
63: Function Add( AComponent : TComponent) : Integer
64: Function Add( AItem, AData : Integer) : Integer
65: Function Add( AItem, AData : Pointer) : Pointer
66: Function Add( AItem, AData : TObject) : TObject
67: Function Add( AObject : TObject) : Integer
68: Function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
69: Function Add( const S : WideString) : Integer
70: Function Add( Image, Mask : TBitmap) : Integer
71: Function Add( Index : LongInt; const Text : string) : LongInt
72: Function Add( Sibling : TTreeNode; const S : string) : TTreeNode
73: Function Add(const S: string): Integer
74: function Add(S: string): Integer;
75: Function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
76: Function ADDCHILD : TFIELDDEF
77: Function AddChild( Index : LongInt; const Text : string) : LongInt
78: Function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
79: Function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
80: Function AddChildObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
81: Function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
82: Function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
83: Function ADDFIELDDEF : TFIELDDEF
84: Function AddFileExtIfNecessary( AFileName, AExt : string) : string
85: Function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
86: Function AddIcon( Image : TIcon) : Integer
87: Function AddImage( Value : TCustomImageList; Index : Integer) : Integer
88: Function ADDINDEXDEF : TINDEXDEF
89: Function AddItem(const Caption:String;const ImageIdx,SelectImageIdx,OverlayImagIdx,
    Indent:Int;Data:Ptr):TComboExItem
```

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

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

```
268: Function BooleanToInteger( const Pb : Boolean) : Integer
269: Function BoolToStr(B: Boolean; UseBoolStrs: Boolean): string)
270: Function BoolToStr1(value : boolean) : string;
271: Function booltoint( aBool : Boolean) : LongInt
272: Function inttobool( aInt : LongInt) : Boolean
273: Function Bounds( ALeft, ATop, AWidth, AHeight : Integer) : TRect
274: function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect)
275: Function BreakApart( BaseString, BreakString : string; StringList : TStrings) : TStrings
276: Function BrightColor( const Color : TColor; const Pct : Single) : TColor
277: Function BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte
278: Function BufferRequest( Length : Integer) : TStream
279: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
280: Function Buttons : PTaskDialogButton
281: Function BytesPerScanline( PixelsPerScanline, BitsPerPixel, Alignment : Longint) : Longint
282: Function BytesToCardinal( const AValue : TIdBytes; const AIndex : Integer) : Cardinal
283: Function BytesToChar( const AValue : TIdBytes; const AIndex : Integer) : Char
284: Function BytesToInt64( const AValue : TIdBytes; const AIndex : Integer) : Int64
285: Function BytesToInteger( const AValue : TIdBytes; const AIndex : Integer) : Integer
286: Function BytesToIPv6( const AValue : TIdBytes; const AIndex : Integer) : TIdIPv6Address
287: Function BytesToShort( const AValue : TIdBytes; const AIndex : Integer) : Short
288: Function BytesToString(ABytes:TIdBytes; AStartIndex:Integer; AMaxCount:Integer): string;
289: Function BytesToStr(const Value: TBytes): String;
290: Function BytesToWord( const AValue : TIdBytes; const AIndex : Integer) : Word
291: Function ByteToBin(Int: Byte): String;
292: Function ByteToCharIndex( S : string; Index : Integer) : Integer
293: function ByteToCharIndex(const S: string; Index: Integer): Integer)
294: Function ByteToCharLen( S : string; MaxLen : Integer) : Integer
295: Function ByteToCharLen(const S: string; MaxLen: Integer): Integer)
296: Function ByteToHex( const AByte : Byte) : string
297: Function ByteToOctal( const AByte : Byte) : string
298: Function ByteType( S : string; Index : Integer) : TMbcsByteType
299: function ByteType(const S: string; Index: Integer): TMbcsByteType)
300: Function CalcTitleRect( Col : TColumn; ARow : Integer; var MasterCol : TColumn) : TRect
301: Function CalculateDFAFingerprint( oStates : TList) : integer
302: function CallTerminateProcs: Boolean)
303: function CANFOCUS:BOOLEAN
304: Function CanLoad( const Ext : string) : Boolean
305: Function CanParse( AWebRequest : TWebRequest) : Boolean
306: Function CanSave( const Ext : string) : Boolean
307: Function CanStart( cChar : char) : boolean
308: Function CaptureScreen : TBitmap;
309: Function CaptureScreen1( Rec : TRect) : TBitmap;
310: Function CardinalToFourChar( ACardinal : LongWord) : string
311: Function CastSoapToNative(Info:PTypeInfo;const SoapData:WideString;NatData:Pointer;IsNull:Boolean): Boolean
312: Function CastSoapToVariant1( SoapInfo : PTypeInfo; const SoapData : WideString) : Variant;
313: Function Ceil( const X : Extended) : Integer
314: Function Ceil16( X : Integer) : Integer
315: Function Ceil4( X : Integer) : Integer
316: Function Ceil8( X : Integer) : Integer
317: Function Ceiling( const X : Extended) : Integer
318: Function CellRect( ACol, ARow : Longint) : TRect
319: Function CelsiusToFahrenheit( const AValue : Double) : Double
320: Function CenterPoint( const Rect : TRect) : TPoint
321: function CenterPoint(const Rect: TRect): TPoint)
322: Function ChangeFileExt( FileName, Extension : string) : string
323: function ChangeFileExt(const FileName: string; const Extension: string): string)
324: Function CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) : Boolean
325: Function CharInSet( const Ch : Char; const testSet: TSysCharSet): Boolean
326: Function CharIsInEOF( const AString : string; ACharPos : Integer) : Boolean
327: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : String) : Boolean
328: Function CharLength( S : String; Index : Integer) : Integer
329: Function CharRange( const AMin, AMax : Char) : String
330: function CharsetToIdent(Charset: Longint; var Ident: string): Boolean)
331: Function CharToBin(vChr: Char): String;
332: Function CharNext(lpsz: PChar): PChar; stdcall;
333: Function CharToByteIndex( S : string; Index : Integer) : Integer
334: Function CharToByteIndex(const S: string; Index: Integer): Integer)
335: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
336: function CharToByteLen(const S: string; MaxLen: Integer): Integer)
337: Function CharToHex(const APrefix : String; const cc : Char) : shortstring;
338: function CharToHexStr(Value: char): string);
339: function CharToOem(ins, outs: PChar):boolean;
340: function CharToUniCode(Value: Char): string;
341: Function CheckMenuDropdown : Boolean
342: Function CheckMessages : longint
343: Function CheckBox: string;
344: Function CheckOpen( Status : DBIResult) : Boolean
345: Function CheckPassword( const APassword : String) : Boolean
346: Function CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt
347: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
348: function CheckSynchronize(Timeout: Integer): Boolean
349: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
350: Function CheckCom(AComNumber: Integer): Integer;');
351: Function CheckLPT1: string;');
352: function ChrA(const a: byte): char;
353: Function ClassIDToProgID(const ClassID: TGUID): string;
354: Function ClassNameIs(const Name: string): Boolean
355: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
356: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
1754: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1755: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1756: Function KeepAlive : Boolean
1757: Function KeysToShiftState(Keys: Word): TShiftState;
1758: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1759: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1760: Function KeyboardStateToShiftState: TShiftState; overload;
1761: Function Languages : TLanguages
1762: Function Last : TClass
1763: Function Last : TComponent
1764: Function Last : TObject
1765: Function LastDelimiter( Delimiters, S : string) : Integer
1766: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1767: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1768: Function Latitude2WGS84(lat: double): double;
1769: Function LCM(m,n:longint):longint;
1770: Function LCMJ( const X, Y : Cardinal) : Cardinal
1771: Function Ldexp( const X : Extended; const P : Integer) : Extended
1772: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1773: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1774: function Length: Integer;
1775: Procedure LetFileList(FileList: TStringlist; apath: string);
1776: function lengthmp3(mp3path: string):integer;
1777: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1778: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1779: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1780: function LineStart(Buffer, BufPos: PChar): PChar
1781: function LineStart(Buffer, BufPos: PChar): PChar)
1782: function ListSeparator: char;
1783: function Ln(x: Extended): Extended;
1784: Function LnXP1( const X : Extended) : Extended
1785: function Lo(vdat: word): byte;
1786: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1787: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1788: Function LoadFileAsString( const FileName : string) : string
1789: Function LoadFromFile( const FileName : string) : TBitmapLoader
1790: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1791: Function LoadPackage(const Name: string): HMODULE
1792: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1793: Function LoadStr( Ident : Integer) : string
1794: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1795: Function LoadWideStr( Ident : Integer) : WideString
1796: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1797: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1798: Function LockServer( fLock : LongBool) : HResult
1799: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1800: Function Log( const X : Extended) : Extended
1801: Function Log10( const X : Extended) : Extended
1802: Function Log2( const X : Extended) : Extended
1803: function LogBase10(X: Float): Float;
1804: Function LogBase2(X: Float): Float;
1805: Function LogBaseN(Base, X: Float): Float;
1806: Function LogN( const Base, X : Extended) : Extended
1807: Function LogOffOS : Boolean
1808: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1809: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1810: Function LongDateFormat: string;
1811: Function LongTimeFormat: string;
1812: Function LongWordToFourChar( ACardinal : LongWord) : string
1813: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1814: Function LookupName( const name : string) : TInAddr
1815: Function LookupService( const service : string) : Integer
1816: function Low: Int64;
1817: Function LowerCase( S : string) : string
1818: Function Lowercase(s : AnyString) : AnyString;
1819: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1820: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1821: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1822: function MainInstance: longword
1823: function MainThreadID: longword
1824: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1825: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1826: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1827: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1828: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1829: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1830: Function MakeFile(const FileName: string): integer)');
1831: function MakeLong(A, B: Word): Longint)
1832: Function MakeTempFilename( const APath : String) : string
1833: Function MakeValidFileName( const Str : string) : string
1834: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1835: function MakeWord(A, B: Byte): Word)
1836: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1837: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1838: Function MapValues( Mapping : string; Value : string) : string
1839: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1840: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1841: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
```

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

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

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

```
2096: Function PointAssign( const X, Y : Integer) : TPoint
2097: Function PointDist( const P1, P2 : TPoint) : Double;
2098: function PointDist(const P1,P2: TFloatPoint): Double;
2099: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2100: function PointDist2(const P1,P2: TPoint): Double;
2101: Function PointEqual( const P1, P2 : TPoint) : Boolean
2102: Function PointIsNull( const P : TPoint) : Boolean
2103: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2104: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2105: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2106: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2107: Function Pop : Pointer
2108: Function Pop : TObject
2109: Function PopnStdDev( const Data : array of Double) : Extended
2110: Function PopnVariance( const Data : array of Double) : Extended
2111: Function PopulationVariance( const X : TDynFloatArray) : Float
2112: function Pos(SubStr, S: AnyString): Longint;
2113: Function PosEqual( const Rect : TRect) : Boolean
2114: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2115: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2116: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2117: Function Post1( AURL : string; const ASource : TStrings) : string;
2118: Function Post2( AURL : string; const ASource : TStream) : string;
2119: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2120: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2121: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2122: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2123: Function Power( const Base, Exponent : Extended) : Extended
2124: Function PowerBig(aval, n:integer): string;
2125: Function PowerIntJ( const X : Float; N : Integer) : Float;
2126: Function PowerJ( const Base, Exponent : Float) : Float;
2127: Function PowerOffOS : Boolean
2128: Function PreformatDateString( Ps : string) : string
2129: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2130: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2131: Function Printer : TPrinter
2132: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2133: Function ProcessResponse : TIdHTTPWhatsNext
2134: Function ProduceContent : string
2135: Function ProduceContentFromStream( Stream : TStream) : string
2136: Function ProduceContentFromString( const S : string) : string
2137: Function ProgIDToClassID(const ProgID: string): TGUID;
2138: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2139: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2140: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2141: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2142: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2143: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2144: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2145: Function Push( AItem : Pointer) : Pointer
2146: Function Push( AObject : TObject) : TObject
2147: Function Put1( AURL : string; const ASource : TStream) : string;
2148: Function Pythagoras( const X, Y : Extended) : Extended
2149: Function queryDLLInterface( var queryList : TStringList) : TStringList
2150: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2151: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2152: Function queryPerformanceCounter2(mse: int64): int64;
2153: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2154: //Function QueryPerformanceFrequency(mse: int64): boolean;
2155: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2156: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2157: Procedure QueryPerformanceCounter1(var aC: Int64);
2158: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2159: Function Quote( const ACommand : String) : SmallInt
2160: Function QuotedStr( S : string) : string
2161: Function RadToCycle( const Radians : Extended) : Extended
2162: Function RadToDeg( const Radians : Extended) : Extended
2163: Function RadToDeg( const Value : Extended) : Extended;
2164: Function RadToDeg1( const Value : Double) : Double;
2165: Function RadToDeg2( const Value : Single) : Single;
2166: Function RadToGrad( const Radians : Extended) : Extended
2167: Function RadToGrad( const Value : Extended) : Extended;
2168: Function RadToGrad1( const Value : Double) : Double;
2169: Function RadToGrad2( const Value : Single) : Single;
2170: Function RandG( Mean, StdDev : Extended) : Extended
2171: function Random(const ARange: Integer): Integer;
2172: function random2(a: integer): double
2173: function RandomE: Extended;
2174: function RandomF: Extended;
2175: Function RandomFrom( const AValues : array of string) : string;
2176: Function RandomRange( const AFrom, ATo : Integer) : Integer
2177: function randSeed: longint
2178: Function RawToDataColumn( ACol : Integer) : Integer
2179: Function Read : Char
2180: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2181: function Read(Buffer:String;Count:LongInt):LongInt
```

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

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

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

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

```
2529: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2530: Function StartsWith( const str : string; const sub : string) : Boolean
2531: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2532: Function StatusString( StatusCode : Integer) : string
2533: Function StdDev( const Data : array of Double) : Extended
2534: Function Stop : Float
2535: Function StopCount( var Counter : TJclCounter) : Float
2536: Function StoreColumns : Boolean
2537: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2538: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2539: Function StrAlloc( Size : Cardinal) : PChar
2540: function StrAlloc(Size: Cardinal): PChar)
2541: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2542: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2543: Function StrBufSize( Str : PChar) : Cardinal
2544: function StrBufSize(const Str: PChar): Cardinal)
2545: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2546: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2547: Function StrCat( Dest : PChar; Source : PChar) : PChar
2548: function StrCat(Dest: PChar; const Source: PChar): PChar)
2549: Function StrCharLength( Str : PChar) : Integer
2550: Function StrComp( Str1, Str2 : PChar) : Integer
2551: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2552: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2553: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2554: Function Stream_to_AnsiString( Source : TStream) : ansistring
2555: Function Stream_to_Base64( Source : TStream) : ansistring
2556: Function Stream_to_decimalbytes( Source : TStream) : string
2557: Function Stream2WideString( oStream : TStream) : WideString
2558: Function StreamtoAnsiString( Source : TStream) : ansistring
2559: Function StreamToByte( Source : TStream) : string
2560: Function StreamToDecimalbytes( Source : TStream) : string
2561: Function StreamtoOrd( Source : TStream) : string
2562: Function StreamToString( Source : TStream) : string
2563: Function StreamToString2( Source : TStream) : string
2564: Function StreamToString3( Source : TStream) : string
2565: Function StreamToString4( Source : TStream) : string
2566: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2567: Function StrEmpty( const sString : string) : boolean
2568: Function StrEnd( Str : PChar) : PChar
2569: function StrEnd(const Str: PChar): PChar)
2570: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2571: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2572: Function StrGet(var S : String; I : Integer) : Char;
2573: Function StrGet2(S : String; I : Integer) : Char;
2574: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2575: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2576: Function StrHtmlDecode( const AStr : String) : String
2577: Function StrHtmlEncode( const AStr : String) : String
2578: Function StrToBytes(const Value: String): TBytes;
2579: Function StrIComp( Str1, Str2 : PChar) : Integer
2580: Function StringOfChar(c : char;I : longInt) : String;
2581: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2582: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2583: Function StringRefCount(const s: String): integer;
2584: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2585: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2586: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2587: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2588: Function StringToBoolean( const Ps : string) : Boolean
2589: function StringToColor(const S: string): TColor)
2590: function StringToCursor(const S: string): TCursor;
2591: function StringToGUID(const S: string): TGUID)
2592: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2593: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2594: Function StringWidth( S : string) : Integer
2595: Function StrInternetToDateTime( Value : string) : TDateTime
2596: Function StrIsDateTime( const Ps : string) : Boolean
2597: Function StrIsFloatMoney( const Ps : string) : Boolean
2598: Function StrIsInteger( const S : string) : Boolean
2599: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2600: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2601: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2602: Function StrLen( Str : PChar) : Cardinal
2603: function StrLen(const Str: PChar): Cardinal)
2604: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2605: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2606: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2607: Function StrLower( Str : PChar) : PChar
2608: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2609: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2610: Function StrNew( Str : PChar) : PChar
2611: function StrNew(const Str: PChar): PChar)
2612: Function StrNextChar( Str : PChar) : PChar
2613: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2614: Function StrParse( var sString : string; const sDelimiters : string) : string;
2615: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2616: Function StrPas( Str : PChar) : string
2617: function StrPas(const Str: PChar): string)
```

```
2618: Function StrPCopy( Dest : PChar; Source : string) : PChar
2619: function StrPCopy(Dest: PChar; const Source: string): PChar)
2620: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2621: Function StrPos( Str1, Str2 : PChar) : PChar
2622: Function StrScan(const Str: PChar; Chr: Char): PChar)
2623: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2624: Function StrToBcd( const AValue : string) : TBcd
2625: Function StrToBool( S : string) : Boolean
2626: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2627: Function StrToCard( const AStr : String) : Cardinal
2628: Function StrToConv( AText : string; out AType : TConvType) : Double
2629: Function StrToCurr( S : string) : Currency;
2630: function StrToCurr(const S: string): Currency)
2631: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2632: Function StrToDate( S : string) : TDateTime;
2633: function StrToDate(const s: string): TDateTime;
2634: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2635: Function StrToDateTime( S : string) : TDateTime;
2636: function StrToDateTime(const S: string): TDateTime)
2637: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2638: Function StrToDay( const ADay : string) : Byte
2639: Function StrToFloat( S : string) : Extended;
2640: function StrToFloat(s: String): Extended;
2641: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2642: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2643: Function StrToFloat( S : string) : Extended;
2644: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2645: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2646: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2647: Function StrToCurr( S : string) : Currency;
2648: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2649: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2650: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2651: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2652: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2653: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2654: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2655: Function StrToDateTime( S : string) : TDateTime;
2656: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2657: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2658: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2659: Function StrToInt( S : string) : Integer
2660: function StrToInt(s: String): Longint;
2661: Function StrToInt64( S : string) : Int64
2662: function StrToInt64(s: String): int64;
2663: Function StrToInt64Def( S : string; Default : Int64) : Int64
2664: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2665: Function StrToIntDef( S : string; Default : Integer) : Integer
2666: function StrToIntDef(const S: string; Default: Integer): Integer)
2667: function StrToIntDef(s: String; def: Longint): Longint;
2668: Function StrToMonth( const AMonth : string) : Byte
2669: Function StrToTime( S : string) : TDateTime;
2670: function StrToTime(const S: string): TDateTime)
2671: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2672: Function StrToWord( const Value : String) : Word
2673: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2674: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2675: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2676: Function StrUpper( Str : PChar) : PChar
2677: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2678: Function Sum( const Data : array of Double) : Extended
2679: Function SumFloatArray( const B : TDynFloatArray) : Float
2680: Function SumInt( const Data : array of Integer) : Integer
2681: Function SumOfSquares( const Data : array of Double) : Extended
2682: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2683: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2684: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2685: Function Supports( CursorOptions : TCursorOptions) : Boolean
2686: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2687: Function SwapWord(w : word): word)
2688: Function SwapInt(i : integer): integer)
2689: Function SwapLong(L : longint): longint)
2690: Function Swap(i : integer): integer)
2691: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2692: Function SyncTime : Boolean
2693: Function SysErrorMessage( ErrorCode : Integer) : string
2694: function SysErrorMessage(ErrorCode: Integer): string)
2695: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2696: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2697: Function SysStringLen(const S: WideString): Integer; stdcall;
2698: Function TabRect( Index : Integer) : TRect
2699: Function Tan( const X : Extended) : Extended
2700: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2701: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2702: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2703: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
```

```
2704: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2705: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2706: Function TenToY( const Y : Float) : Float
2707: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2708: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2709: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2710: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2711: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2712: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2713: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2714: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2715: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2716: Function TestBits( const Value, Mask : Byte) : Boolean;
2717: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2718: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2719: Function TestBits3( const Value, Mask : Word) : Boolean;
2720: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2721: Function TestBits5( const Value, Mask : Integer) : Boolean;
2722: Function TestBits6( const Value, Mask : Int64) : Boolean;
2723: Function TestFDIVInstruction : Boolean
2724: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2725: Function TextExtent( const Text : string) : TSize
2726: function TextHeight(Text: string): Integer;
2727: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2728: Function TextStartsWith( const S, SubS : string) : Boolean
2729: Function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2730: Function ConvInteger(i : integer):string;
2731: Function IntegerToText(i : integer):string;
2732: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2733: function TextWidth(Text: string): Integer;
2734: Function ThreadCount : integer
2735: function ThousandSeparator: char;
2736: Function Ticks : Cardinal
2737: Function Time : TDateTime
2738: function Time: TDateTime;
2739: function TimeGetTime: int64;
2740: Function TimeOf( const AValue : TDateTime) : TDateTime
2741: function TimeSeparator: char;
2742: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2743: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2744: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2745: Function TimeToStr( DateTime : TDateTime) : string;
2746: function TimeToStr(const DateTime: TDateTime): string;
2747: Function TimeZoneBias : TDateTime
2748: Function ToCommon( const AValue : Double) : Double
2749: function ToCommon(const AValue: Double): Double;
2750: Function Today : TDateTime
2751: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2752: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2753: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2754: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2755: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2756: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2757: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2758: function TokenComponentIdent:String
2759: Function TokenFloat : Extended
2760: function TokenFloat:Extended
2761: Function TokenInt : Longint
2762: function TokenInt:LongInt
2763: Function TokenString : string
2764: function TokenString:String
2765: Function TokenSymbolIs( const S : string) : Boolean
2766: function TokenSymbolIs(S:String):Boolean
2767: Function Tomorrow : TDateTime
2768: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2769: Function ToString : string
2770: Function TotalVariance( const Data : array of Double) : Extended
2771: Function Trace2( AURL : string) : string;
2772: Function TrackMenu( Button : TToolButton) : Boolean
2773: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2774: Function TranslateURI( const URI : string) : string
2775: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2776: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2777: Function Trim( S : string) : string;
2778: Function Trim( S : WideString) : WideString;
2779: Function Trim(s : AnyString) : AnyString;
2780: Function TrimAllOf( ATrim, AText : String) : String
2781: Function TrimLeft( S : string) : string;
2782: Function TrimLeft( S : WideString) : WideString;
2783: function TrimLeft(const S: string): string)
2784: Function TrimRight( S : string) : string;
2785: Function TrimRight( S : WideString) : WideString;
2786: function TrimRight(const S: string): string)
2787: function TrueBoolStrs: array of string
2788: Function Trunc(e : Extended) : Longint;
2789: Function Trunc64(e: extended): Int64;
```

```
2790: Function TruncPower( const Base, Exponent : Float) : Float
2791: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2792: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2793: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2794: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2795: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2796: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
      AValue:TDateTime):Boolean
2797: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2798: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
      AVal:TDateTime):Bool
2799: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2800: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2801: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2802: Function TryLock : Boolean
2803: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2804: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
2805: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2806: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2807: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2808: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2809: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2810: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2811: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2812: function TryStrToBool(const S: string; out Value: Boolean): Boolean;
2813: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2814: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2815: Function TwoToY( const Y : Float) : Float
2816: Function UCS4StringToWideString( const S : UCS4String) : WideString
2817: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2818: function Unassigned: Variant;
2819: Function UndoLastChange( FollowChange : Boolean) : Boolean
2820: function UniCodeToStr(Value: string): string;
2821: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2822: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2823: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2824: Function UnixPathToDosPath( const Path : string) : string
2825: Function UnixToDateTime( const AValue : Int64) : TDateTime
2826: function UnixToDateTime(U: Int64): TDateTime;
2827: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2828: Function UnlockResource( ResData : HGLOBAL) : LongBool
2829: Function UnlockVolume( var Handle : THandle) : Boolean
2830: Function UnMaskString( Mask, Value : String) : String
2831: function UpCase(ch : Char ) : Char;
2832: Function UpCaseFirst( const AStr : string) : string
2833: Function UpCaseFirstWord( const AStr : string) : string
2834: Function UpdateAction( Action : TBasicAction) : Boolean
2835: Function UpdateKind : TUpdateKind
2836: Function UPDATESTATUS : TUPDATESTATUS
2837: Function UpperCase( S : string) : string
2838: Function Uppercase(s : AnyString) : AnyString;
2839: Function URLDecode( ASrc : string) : string
2840: Function URLEncode( const ASrc : string) : string
2841: Function UseRightToLeftAlignment : Boolean
2842: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2843: Function UseRightToLeftReading : Boolean
2844: Function UTF8CharLength( Lead : Char) : Integer
2845: Function UTF8CharSize( Lead : Char) : Integer
2846: Function UTF8Decode( const S : UTF8String) : WideString
2847: Function UTF8Encode( const WS : WideString) : UTF8String
2848: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2849: Function Utf8ToAnsi( const S : UTF8String) : string
2850: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2851: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2852: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2853: Function ValidParentForm(control: TControl): TForm
2854: Function Value : Variant
2855: Function ValueExists( const Section, Ident : string) : Boolean
2856: Function ValueOf( const Key : string) : Integer
2857: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2858: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2859: Function VarArrayFromStrings( Strings : TStrings) : Variant
2860: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2861: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2862: Function VarFMTBcd : TVarType
2863: Function VarFMTBcdCreate1 : Variant;
2864: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2865: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2866: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2867: Function Variance( const Data : array of Double) : Extended
2868: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2869: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2870: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2871: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2872: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2873: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2874: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2875: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
```

```
2876: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2877: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2878: Function VariantNeg( const V1 : Variant) : Variant
2879: Function VariantNot( const V1 : Variant) : Variant
2880: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2881: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2882: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2883: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2884: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2885: function VarIsEmpty(const V: Variant): Boolean;
2886: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2887: function VarIsNull(const V: Variant): Boolean;
2888: Function VarToBcd( const AValue : Variant) : TBcd
2889: function VarType(const V: Variant): TVarType;
2890: Function VarType( const V : Variant) : TVarType
2891: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2892: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2893: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2894: Function VarIsByRef( const V : Variant) : Boolean
2895: Function VarIsEmpty( const V : Variant) : Boolean
2896: Procedure VarCheckEmpty( const V : Variant)
2897: Function VarIsNull( const V : Variant) : Boolean
2898: Function VarIsClear( const V : Variant) : Boolean
2899: Function VarIsCustom( const V : Variant) : Boolean
2900: Function VarIsOrdinal( const V : Variant) : Boolean
2901: Function VarIsFloat( const V : Variant) : Boolean
2902: Function VarIsNumeric( const V : Variant) : Boolean
2903: Function VarIsStr( const V : Variant) : Boolean
2904: Function VarToStr( const V : Variant) : string
2905: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2906: Function VarToWideStr( const V : Variant) : WideString
2907: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2908: Function VarToDateTime( const V : Variant) : TDateTime
2909: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2910: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2911: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2912: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2913: Function VarSameValue( const A, B : Variant) : Boolean
2914: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2915: Function VarIsEmptyParam( const V : Variant) : Boolean
2916: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2917: Function VarIsError1( const V : Variant) : Boolean;
2918: Function VarAsError( AResult : HRESULT) : Variant
2919: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2920: Function VarIsArray( const A : Variant) : Boolean;
2921: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2922: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2923: Function VarArrayOf( const Values : array of Variant) : Variant
2924: Function VarArrayRef( const A : Variant) : Variant
2925: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2926: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2927: Function VarArrayDimCount( const A : Variant) : Integer
2928: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2929: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2930: Function VarArrayLock( const A : Variant) : ___Pointer
2931: Procedure VarArrayUnlock( const A : Variant)
2932: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2933: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2934: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2935: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2936: Function Unassigned : Variant
2937: Function Null : Variant
2938: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2939: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2940: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2941: function VectorDot(const V1,V2: TFloatPoint): Double;
2942: Function VectorLengthSqr( const V : TFloatPoint) : Double
2943: function VectorLengthSqr(const V: TFloatPoint): Double;
2944: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2945: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2946: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2947: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2948: Function Verify( AUserName : String) : String
2949: Function Versine( X : Float) : Float
2950: function VersionCheck: boolean;
2951: function VersionCheckAct: string;
2952: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2953: Function VersionLanguageName( const LangId : Word) : string
2954: Function VersionResourceAvailable( const FileName : string) : Boolean
2955: Function Visible : Boolean
2956: function VolumeID(DriveChar: Char): string
2957: Function WaitFor( const AString : string) : string
2958: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2959: Function WaitFor1 : TWaitResult;
2960: Function WaitForData( Timeout : Longint) : Boolean
2961: Function WebColorNameToColor( WebColorName : string) : TColor
2962: Function WebColorStrToColor( WebColor : string) : TColor
2963: Function WebColorToRGB( WebColor : Integer) : Integer
2964: Function wGet(aURL, afile: string): boolean;'
```

```
2965: Function wGet2(aURL, afile: string): boolean;'  //without file open
2966: Function wGetX(aURL, afile: string): boolean;
2967: Function wGetX2(aURL, afile: string): boolean;'  //without file open
2968: Function WebGet(aURL, afile: string): boolean;'
2969: Function WebExists: boolean;   //alias to isinternet
2970: Function WeekOf( const AValue : TDateTime) : Word
2971: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2972: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2973: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2974: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2975: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2976: Function WeeksInAYear( const AYear : Word) : Word
2977: Function WeeksInYear( const AValue : TDateTime) : Word
2978: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2979: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2980: Function WideCat( const x, y : WideString) : WideString
2981: Function WideCompareStr( S1, S2 : WideString) : Integer
2982: function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
2983: Function WideCompareText( S1, S2 : WideString) : Integer
2984: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
2985: Function WideCopy( const src : WideString; index, count : Integer) : WideString
2986: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
2987: Function WideEqual( const x, y : WideString) : Boolean
2988: Function WideFormat(const Format: WideString; const Args: array of const): WideString)
2989: Function WideGreater( const x, y : WideString) : Boolean
2990: Function WideLength( const src : WideString) : Integer
2991: Function WideLess( const x, y : WideString) : Boolean
2992: Function WideLowerCase( S : WideString) : WideString
2993: function WideLowerCase(const S: WideString): WideString)
2994: Function WidePos( const src, sub : WideString) : Integer
2995: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
2996: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
2997: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
2998: Function WideSameStr( S1, S2 : WideString) : Boolean
2999: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
3000: Function WideSameText( S1, S2 : WideString) : Boolean
3001: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
3002: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
3003: Function WideStringToUCS4String( const S : WideString) : UCS4String
3004: Function WideUpperCase( S : WideString) : WideString
3005: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
3006: function Win32Check(RetVal: boolean): boolean)
3007: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
3008: Function Win32RestoreFile( const FileName : string) : Boolean
3009: Function Win32Type : TIdWin32Type
3010: Function WinColor( const Color32 : TColor32) : TColor
3011: function winexec(FileName: pchar; showCmd: integer): integer;
3012: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3013: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3014: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
3015: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
3016: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
3017: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
3018: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
3019: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
3020: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
3021: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
3022: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
3023: Function WordToStr( const Value : Word) : String
3024: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
3025: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
3026: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
3027: Function WorkArea : Integer
3028: Function WrapText( Line : string; MaxCol : Integer) : string;
3029: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
3030: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
3031: function Write(Buffer:String;Count:LongInt):LongInt
3032: Function WriteClient( var Buffer, Count : Integer) : Integer
3033: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
3034: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
3035: Function WriteString( const AString : string) : Boolean
3036: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
3037: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
3038: Function wsprintf( Output : PChar; Format : PChar) : Integer
3039: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
3040: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
3041: Function XorDecode( const Key, Source : string) : string
3042: Function XorEncode( const Key, Source : string) : string
3043: Function XorString( const Key, Src : ShortString) : ShortString
3044: Function Yield : Bool
3045: Function YearOf( const AValue : TDateTime) : Word
3046: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3047: Function YearSpan( const ANow, AThen : TDateTime) : Double
3048: Function Yesterday : TDateTime
3049: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3050: Function( const Name : string; Proc : TUserFunction)
3051: Function using Special_Scholz from 3.8.5.0
3052: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3053: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
```

```
3054: Function FloatToTime2Dec(value:Extended):Extended;
3055: Function MinToStd(value:Extended):Extended;
3056: Function MinToStdAsString(value:Extended):String;
3057: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3058: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3059: Function Round2Dec (zahl:Extended):Extended;
3060: Function GetAngle(x,y:Extended):Double;
3061: Function AddAngle(a1,a2:Double):Double;
3062:
3063: ****************************************************************
3064: unit uPSI_StText;
3065: ****************************************************************
3066: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3067: Function TextFileSize( var F : TextFile) : LongInt
3068: Function TextPos( var F : TextFile) : LongInt
3069: Function TextFlush( var F : TextFile) : Boolean
3070:
3071: ****************************************************************
3072: from  JvVCLUtils;
3073: ****************************************************************
3074: { Windows resources (bitmaps and icons) VCL-oriented routines }
3075: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3076: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
     DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3077: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
     Bitmap: TBitmap; TransparentColor: TColor);
3078: function MakeBitmap(ResID: PChar): TBitmap;
3079: function MakeBitmapID(ResID: Word): TBitmap;
3080: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3081: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3082: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3083: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3084:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3085: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3086: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3087: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3088: {$IFDEF WIN32}
3089: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3090:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3091: {$ENDIF}
3092: function MakeIcon(ResID: PChar): TIcon;
3093: function MakeIconID(ResID: Word): TIcon;
3094: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3095: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3096: {$IFDEF WIN32}
3097: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3098: {$ENDIF}
3099: { Service routines }
3100: procedure NotImplemented;
3101: procedure ResourceNotFound(ResID: PChar);
3102: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3103: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3104: function PaletteColor(Color: TColor): Longint;
3105: function WidthOf(R: TRect): Integer;
3106: function HeightOf(R: TRect): Integer;
3107: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3108: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3109: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3110: procedure Delay(MSecs: Longint);
3111: procedure CenterControl(Control: TControl);
3112: Function PaletteEntries( Palette : HPALETTE) : Integer
3113: Function WindowClassName( Wnd : HWND) : string
3114: Function ScreenWorkArea : TRect
3115: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3116: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3117: Procedure ActivateWindow( Wnd : HWND)
3118: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3119: Procedure CenterWindow( Wnd : HWND)
3120: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3121: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3122: Function DialogsToPixelsX( Dlgs : Word) : Word
3123: Function DialogsToPixelsY( Dlgs : Word) : Word
3124: Function PixelsToDialogsX( Pixs : Word) : Word
3125: Function PixelsToDialogsY( Pixs : Word) : Word
3126: {$IFDEF WIN32}
3127: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3128: function MakeVariant(const Values: array of Variant): Variant;
3129: {$ENDIF}
3130: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3131: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3132: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3133: {$IFDEF CBUILDER}
3134: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3135: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3136: {$ELSE}
3137: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3138: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3139: {$ENDIF CBUILDER}
3140: function IsForegroundTask: Boolean;
```

```
3141: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3142: function GetAveCharSize(Canvas: TCanvas): TPoint;
3143: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3144: procedure FreeUnusedOle;
3145: procedure Beep;
3146: function GetWindowsVersionJ: string;
3147: function LoadDLL(const LibName: string): THandle;
3148: function RegisterServer(const ModuleName: string): Boolean;
3149: {$IFNDEF WIN32}
3150: function IsLibrary: Boolean;
3151: {$ENDIF}
3152: { Gradient filling routine }
3153: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3154: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
      TFillDirection; Colors: Byte);
3155: { String routines }
3156: function GetEnvVar(const VarName: string): string;
3157: function AnsiUpperFirstChar(const S: string): string;
3158: function StringToPChar(var S: string): PChar;
3159: function StrPAlloc(const S: string): PChar;
3160: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3161: function DropT(const S: string): string;
3162: { Memory routines }
3163: function AllocMemo(Size: Longint): Pointer;
3164: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3165: procedure FreeMemo(var fpBlock: Pointer);
3166: function GetMemoSize(fpBlock: Pointer): Longint;
3167: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3168: {$IFNDEF COMPILER5_UP}
3169: procedure FreeAndNil(var Obj);
3170: {$ENDIF}
3171: // from PNGLoader
3172: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3173: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3174: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3175: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3176: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
      TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3177:   Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3178:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3179:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3180:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3181:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3182:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3183:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3184:  Procedure SetImeName( Name : TImeName)
3185:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3186:  Function Imm32GetContext( hWnd : HWND) : HIMC
3187:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3188:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
3189:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3190:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3191: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3192:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3193:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3194:  Function Imm32IsIME( hKl : longword) : Boolean
3195:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3196:  Procedure DragDone( Drop : Boolean)
3197:
3198:
3199: //**************************************added  from jvjvclutils
3200: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3201: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3202: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3203: function IsPositiveResult(Value: TModalResult): Boolean;
3204: function IsNegativeResult(Value: TModalResult): Boolean;
3205: function IsAbortResult(const Value: TModalResult): Boolean;
3206: function StripAllFromResult(const Value: TModalResult): TModalResult;
3207: // returns either BrightColor or DarkColor depending on the luminance of AColor
3208: // This function gives the same result (AFAIK) as the function used in Windows to
3209: // calculate the desktop icon text color based on the desktop background color
3210: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3211: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3212:
3213: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3214:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3215:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3216:   var LinkName: string; Scale: Integer = 100); overload;
3217: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3218:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3219:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3220:   var LinkName: string; Scale: Integer = 100); overload;
3221: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3222:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3223: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3224:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3225:   Scale: Integer = 100): string;
3226: function HTMLPlainText(const Text: string): string;
3227: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
```

```
3228:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3229: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3230:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3231: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3232: function HTMLPrepareText(const Text: string): string;
3233:
3234: ****************************************** uPSI_JvAppUtils;
3235: Function GetDefaultSection( Component : TComponent) : string
3236: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3237: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3238: Function GetDefaultIniName : string
3239:  //'OnGetDefaultIniName','TOnGetDefaultIniName);
3240: Function GetDefaultIniRegKey : string
3241: Function FindForm( FormClass : TFormClass) : TForm
3242: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3243: Function ShowDialog( FormClass : TFormClass) : Boolean
3244:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3245: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3246: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3247: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3248: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3249: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3250: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3251: Function StrToIniStr( const Str : string) : string
3252: Function IniStrToStr( const Str : string) : string
3253: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3254: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3255: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3256: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3257: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3258: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
3259: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3260: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3261: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3262: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3263: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3264: Procedure AppTaskbarIcons( AppOnly : Boolean)
3265: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3266: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3267: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3268: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3269: ****************************************** uPSI_JvDBUtils;
3270: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3271: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3272: Procedure RefreshQuery( Query : TDataSet)
3273: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3274: Function DataSetSectionName( DataSet : TDataSet) : string
3275: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string)
3276: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3277: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
      Options: TLocateOptions) : Boolean
3278: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3279: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3280: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3281: Function ConfirmDelete : Boolean
3282: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3283: Procedure CheckRequiredField( Field : TField)
3284: Procedure CheckRequiredFields( const Fields : array of TField)
3285: Function DateToSQL( Value : TDateTime) : string
3286: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3287: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3288: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
      HighEmpty:Double;Inclusive:Bool):string
3289: Function StrMaskSQL( const Value : string) : string
3290: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3291: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3292: Procedure _DBError( const Msg : string)
3293:  Const('TrueExpr','String '0=0
3294:  Const('sdfStandard16','String '''''''mm''/''dd''/''yyyy''''''
3295:  Const('sdfStandard32','String ''''''dd/mm/yyyy''''''
3296:  Const('sdfOracle','String '"TO_DATE(''"dd/mm/yyyy"'', ''DD/MM/YYYY'')"
3297:  Const('sdfInterbase','String '"CAST(''"mm"/"dd"/"yyyy"'' AS DATE)"
3298:  Const('sdfMSSQL','String '"CONVERT(datetime, ''"mm"/"dd"/"yyyy"'', 103)"
3299:  AddTypeS('Largeint', 'Longint
3300:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3301:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3302:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3303:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3304:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3305: (*----------------------------------------------------------------------------*)
3306: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3307: begin
3308:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3309:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3310:  Function JIniReadString( const FileName, Section, Line : string) : string
3311:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3312:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3313:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3314:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
```

```
3315:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3316: end;
3317:
3318: (* === compile-time registration functions === *)
3319: (*--------------------------------------------------------------------------*)
3320: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3321: begin
3322:  'UnixTimeStart','LongInt'( 25569);
3323:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3324:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3325:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3326:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3327:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3328:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3329:  Function DayOfDate( const DateTime : TDateTime) : Integer
3330:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3331:  Function YearOfDate( const DateTime : TDateTime) : Integer
3332:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3333:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3334:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3335:  Function HourOfTime( const DateTime : TDateTime) : Integer
3336:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3337:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3338:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3339:  Function IsISOLongYear( const Year : Word) : Boolean;
3340:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3341:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3342:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3343:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3344:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3345:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3346:  Function JIsLeapYear( const Year : Integer) : Boolean;
3347:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3348:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3349:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3350:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3351:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3352:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3353:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3354:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3355:  Function HoursToMSecs( Hours : Integer) : Integer
3356:  Function MinutesToMSecs( Minutes : Integer) : Integer
3357:  Function SecondsToMSecs( Seconds : Integer) : Integer
3358:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3359:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3360:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3361:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3362:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3363:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3364:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3365:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3366:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3367:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3368:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3369:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3370:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3371:  Function DosDateTimeToStr( DateTime : Integer) : string
3372:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3373:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3374:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3375:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3376:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3377:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3378:  Function FileTimeToStr( const FileTime : TFileTime) : string
3379:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3380:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3381:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3382:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3383:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3384:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3385:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3386:   TJclUnixTime32', 'Longword
3387:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3388:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3389:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3390:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3391:  Function JNullStamp : TTimeStamp
3392:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3393:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3394:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3395:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3396:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3397:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3398:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3399:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3400:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3401:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3402:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3403:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
```

```
3404:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3405:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3406:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3407:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3408:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3409:   FindClass('TOBJECT'),'EJclDateTimeError
3410: end;
3411:
3412: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3413: begin
3414:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3415:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3416:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3417:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3418:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3419:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3420:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3421:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3422:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3423:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3424:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3425:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3426:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3427:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3428:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3429:  Function AbortShutDown : Boolean;
3430:  Function AbortShutDown1( const MachineName : string) : Boolean;
3431:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3432:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3433:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3434:   FindClass('TOBJECT'),'EJclCreateProcessError
3435:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3436:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
     Environment:PChar);
3437:   // with Add(EJclCreateProcessError) do
3438:  end;
3439:
3440:
3441: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3442: begin
3443:   //'AnsiSigns','Set').SetSet(['-', '+']);
3444:  'C1_UPPER','LongWord( $0001);
3445:  'C1_LOWER','LongWord( $0002);
3446:  'C1_DIGIT','LongWord').SetUInt( $0004);
3447:  'C1_SPACE','LongWord').SetUInt( $0008);
3448:  'C1_PUNCT','LongWord').SetUInt( $0010);
3449:  'C1_CNTRL','LongWord').SetUInt( $0020);
3450:  'C1_BLANK','LongWord').SetUInt( $0040);
3451:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3452:  'C1_ALPHA','LongWord').SetUInt( $0100);
3453:   AnsiChar', 'Char
3454:  Function StrIsAlpha( const S : AnsiString) : Boolean
3455:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3456:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3457:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3458:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3459:  Function StrIsDigit( const S : AnsiString) : Boolean
3460:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3461:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3462:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3463:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3464:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3465:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3466:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3467:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3468:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3469:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3470:  Function StrEscapedToString( const S : AnsiString) : AnsiString
3471:  Function JStrLower( const S : AnsiString) : AnsiString
3472:  Procedure StrLowerInPlace( var S : AnsiString)
3473:  ///Procedure StrLowerBuff( S : PAnsiChar)
3474:  Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3475:  Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3476:  Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3477:  Function StrProper( const S : AnsiString) : AnsiString
3478:  //Procedure StrProperBuff( S : PAnsiChar)
3479:  Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3480:  Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3481:  Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3482:  Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3483:  Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3484:  Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3485:  Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3486:  Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3487:  Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3488:  Function StrReverse( const S : AnsiString) : AnsiString
3489:  Procedure StrReverseInPlace( var S : AnsiString)
3490:  Function StrSingleQuote( const S : AnsiString) : AnsiString
3491:  Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
```

```
3492:  Function StrStringToEscaped( const S : AnsiString) : AnsiString
3493:  Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3494:  Function StrToHex( const Source : AnsiString) : AnsiString
3495:  Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3496:  Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3497:  Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3498:  Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3499:  Function StrTrimQuotes( const S : AnsiString) : AnsiString
3500:  Function JStrUpper( const S : AnsiString) : AnsiString
3501:  Procedure StrUpperInPlace( var S : AnsiString)
3502:  //Procedure StrUpperBuff( S : PAnsiChar)
3503:  Function StrOemToAnsi( const S : AnsiString) : AnsiString
3504:  Function StrAnsiToOem( const S : AnsiString) : AnsiString
3505:  Procedure StrAddRef( var S : AnsiString)
3506:  Function StrAllocSize( const S : AnsiString) : Longint
3507:  Procedure StrDecRef( var S : AnsiString)
3508:  //Function StrLen( S : PAnsiChar) : Integer
3509:  Function StrLength( const S : AnsiString) : Longint
3510:  Function StrRefCount( const S : AnsiString) : Longint
3511:  Procedure StrResetLength( var S : AnsiString)
3512:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3513:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3514:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3515:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3516:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3517:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3518:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3519:  Function StrFillChar(const C: Char; Count: Integer): string)');
3520:  Function IntFillChar(const I: Integer; Count: Integer): string)');
3521:  Function ByteFillChar(const B: Byte; Count: Integer): string)');
3522:  Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;');
3523:  Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3524:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3525:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3526:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3527:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
3528:  Function StrIPos( const SubStr, S : AnsiString) : Integer
3529:  Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3530:  Function StrLastPos( const SubStr, S : AnsiString) : Integer
3531:  Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3532:  Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3533:  Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3534:  Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3535:  Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3536:  Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3537:  //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3538:  //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3539:  Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3540:  Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3541:  Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3542:  Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
3543:  Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3544:  Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3545:  Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3546:  Function CharIsAlpha( const C : AnsiChar) : Boolean
3547:  Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3548:  Function CharIsBlank( const C : AnsiChar) : Boolean
3549:  Function CharIsControl( const C : AnsiChar) : Boolean
3550:  Function CharIsDelete( const C : AnsiChar) : Boolean
3551:  Function CharIsDigit( const C : AnsiChar) : Boolean
3552:  Function CharIsLower( const C : AnsiChar) : Boolean
3553:  Function CharIsNumberChar( const C : AnsiChar) : Boolean
3554:  Function CharIsPrintable( const C : AnsiChar) : Boolean
3555:  Function CharIsPunctuation( const C : AnsiChar) : Boolean
3556:  Function CharIsReturn( const C : AnsiChar) : Boolean
3557:  Function CharIsSpace( const C : AnsiChar) : Boolean
3558:  Function CharIsUpper( const C : AnsiChar) : Boolean
3559:  Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3560:  Function CharType( const C : AnsiChar) : Word
3561:  Function CharHex( const C : AnsiChar) : Byte
3562:  Function CharLower( const C : AnsiChar) : AnsiChar
3563:  Function CharUpper( const C : AnsiChar) : AnsiChar
3564:  Function CharToggleCase( const C : AnsiChar) : AnsiChar
3565:  Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3566:  Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3567:  Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer) : Integer
3568:  Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar) : Integer
3569:  Procedure StrIToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3570:  Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3571:  Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3572:  Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)
3573:  Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)
3574:  Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)
3575:  Function AddStringToStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3576:  Function BooleanToStr( B : Boolean) : AnsiString
3577:  Function FileToString( const FileName : AnsiString) : AnsiString
3578:  Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean)
3579:  Function StrToken( var S : AnsiString; Separator : AnsiChar) : AnsiString
3580:  Procedure StrTokens( const S : AnsiString; const List : TStrings)
```

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

```
3664: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3665: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3666: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3667: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3668: function ValidDate(ADate: TDateTime): Boolean;
3669: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3670: function MonthsBetween(Date1, Date2: TDateTime): Double;
3671: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3672: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3673: function DaysBetween(Date1, Date2: TDateTime): Longint;
3674: { The same as previous but if Date2 < Date1 result = 0 }
3675: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3676: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3677: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3678: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3679: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3680: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3681: { String to date conversions }
3682: function GetDateOrder(const DateFormat: string): TDateOrder;
3683: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3684: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3685: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3686: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3687: function DefDateFormat(FourDigitYear: Boolean): string;
3688: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3689: ------------------------------------------------------------------------------
3690: ******************************** JvUtils;*****************************
3691: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3692: function GetWordOnPos(const S: string; const P: Integer): string;
3693: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3694: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3695: { SubStr returns substring from string, S, separated with Separator string}
3696: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3697: { SubStrEnd same to previous function but Index numerated from the end of string }
3698: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3699: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3700: function SubWord(P: PChar; var P2: PChar): string;
3701: { NumberByWord returns the text representation of
3702:   the number, N, in normal russian language. Was typed from Monitor magazine }
3703: function NumberByWord(const N: Longint): string;
3704: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3705:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3706: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3707: { GetXYByPos is same to previous function, but returns X position in line too}
3708: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3709: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3710: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3711: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3712: function ConcatSep(const S, S2, Separator: string): string;
3713: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3714: function ConcatLeftSep(const S, S2, Separator: string): string;
3715: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3716: function MinimizeString(const S: string; const MaxLen: Integer): string;
3717: { Next 4 function for russian chars transliterating.
3718:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3719: procedure Dos2Win(var S: string);
3720: procedure Win2Dos(var S: string);
3721: function Dos2WinRes(const S: string): string;
3722: function Win2DosRes(const S: string): string;
3723: function Win2Koi(const S: string): string;
3724: { Spaces returns string consists on N space chars }
3725: function Spaces(const N: Integer): string;
3726: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3727: function AddSpaces(const S: string; const N: Integer): string;
3728: { function LastDate for russian users only } { returns date relative to current date: '' }
3729: function LastDate(const Dat: TDateTime): string;
3730: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3731: function CurrencyToStr(const Cur: currency): string;
3732: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3733: function Cmp(const S1, S2: string): Boolean;
3734: { StringCat add S2 string to S1 and returns this string }
3735: function StringCat(var S1: string; S2: string): string;
3736: { HasChar returns True, if Char, Ch, contains in string, S }
3737: function HasChar(const Ch: Char; const S: string): Boolean;
3738: function HasAnyChar(const Chars: string; const S: string): Boolean;
3739: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3740: function CountOfChar(const Ch: Char; const S: string): Integer;
3741: function DefStr(const S: string; Default: string): string;
3742: {**** files routines}
3743: { GetWinDir returns Windows folder name }
3744: function GetWinDir: TFileName;
3745: function GetSysDir: String;
3746: { GetTempDir returns Windows temporary folder name }
3747: function GetTempDir: string;
3748: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3749: function GenTempFileName(FileName: string): string;
3750: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3751: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3752: { ClearDir clears folder Dir }
```

```
3753: function ClearDir(const Dir: string): Boolean;
3754: { DeleteDir clears and than delete folder Dir }
3755: function DeleteDir(const Dir: string): Boolean;
3756: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3757: function FileEquMask(FileName, Mask: TFileName): Boolean;
3758: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3759:   Masks must be separated with comma (';') }
3760: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3761: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3762: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3763: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3764: { FileGetInfo fills SearchRec record for specified file attributes}
3765: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3766: { HasSubFolder returns True, if folder APath contains other folders }
3767: function HasSubFolder(APath: TFileName): Boolean;
3768: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3769: function IsEmptyFolder(APath: TFileName): Boolean;
3770: { AddSlash add slash Char to Dir parameter, if needed }
3771: procedure AddSlash(var Dir: TFileName);
3772: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3773: function AddSlash2(const Dir: TFileName): string;
3774: { AddPath returns FileName with Path, if FileName not contain any path }
3775: function AddPath(const FileName, Path: TFileName): TFileName;
3776: function AddPaths(const PathList, Path: string): string;
3777: function ParentPath(const Path: TFileName): TFileName;
3778: function FindInPath(const FileName, PathList: string): TFileName;
3779: function FindInPaths(const fileName,paths: String): String;
3780: {$IFNDEF BCB1}
3781: { BrowseForFolder displays Browse For Folder dialog }
3782: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3783: {$ENDIF BCB1}
3784: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3785: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3786: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3787: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3788:
3789: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3790: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3791: { HasParam returns True, if program running with specified parameter, Param }
3792: function HasParam(const Param: string): Boolean;
3793: function HasSwitch(const Param: string): Boolean;
3794: function Switch(const Param: string): string;
3795: { ExePath returns ExtractFilePath(ParamStr(0)) }
3796: function ExePath: TFileName;
3797: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3798: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3799: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3800: {**** Graphic routines }
3801: { TTFontSelected returns True, if True Type font is selected in specified device context }
3802: function TTFontSelected(const DC: HDC): Boolean;
3803: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3804: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3805: {**** Windows routines }
3806: { SetWindowTop put window to top without recreating window }
3807: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3808: {**** other routines }
3809: { KeyPressed returns True, if Key VK is now pressed }
3810: function KeyPressed(VK: Integer): Boolean;
3811: procedure SwapInt(var Int1, Int2: Integer);
3812: function IntPower(Base, Exponent: Integer): Integer;
3813: function ChangeTopException(E: TObject): TObject;
3814: function StrToBool(const S: string): Boolean;
3815: {$IFNDEF COMPILER3_UP}
3816: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3817:   Length of MaxLen bytes. The compare operation is controlled by the
3818:   current Windows locale. The return value is the same as for CompareStr. }
3819: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3820: function AnsiStrIComp(S1, S2: PChar): Integer;
3821: {$ENDIF}
3822: function Var2Type(V: Variant; const VarType: Integer): Variant;
3823: function VarToInt(V: Variant): Integer;
3824: function VarToFloat(V: Variant): Double;
3825: { following functions are not documented because they are don't work properly , so don't use them }
3826: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3827: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3828: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3829: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3830: function GetParameter: string;
3831: function GetLongFileName(FileName: string): string;
3832: {* from  FileCtrl}
3833: function DirectoryExists(const Name: string): Boolean;
3834: procedure ForceDirectories(Dir: string);
3835: {# from  FileCtrl}
3836: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3837: function GetComputerID: string;
3838: function GetComputerName: string;
3839: {**** string routines }
```

```
3840: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 function }
3841: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3842: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3843:   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 }
3844: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3845: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3846: function CountOfLines(const S: string): Integer;
3847: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3848: procedure DeleteEmptyLines(Ss: TStrings);
3849: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3850:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3851: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3852: {**** files routines - }
3853: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3854:   Resource can be compressed using MS Compress program}
3855: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3856: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3857: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3858: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3859: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3860: { IniReadSection read section, Section, from ini-file,
3861:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3862:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3863: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3864: { LoadTextFile load text file, FileName, into string }
3865: function LoadTextFile(const FileName: TFileName): string;
3866: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3867: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3868: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3869: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3870: {$IFDEF COMPILER3_UP}
3871: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3872: function TargetFileName(const FileName: TFileName): TFileName;
3873: { return filename ShortCut linked to }
3874: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3875: {$ENDIF COMPILER3_UP}
3876: {**** Graphic routines - }
3877: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3878: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3879: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3880: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3881: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3882: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3883: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3884: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3885: { Cinema draws some visual effect }
3886: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3887: { Roughed fills rect with special 3D pattern }
3888: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3889: { 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 }
3890: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3891: { TextWidth calculate text with for writing using standard desktop font }
3892: function TextWidth(AStr: string): Integer;
3893: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3894: function DefineCursor(Identifer: PChar): TCursor;
3895: {**** other routines - }
3896: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3897: function FindFormByClass(FormClass: TFormClass): TForm;
3898: function FindFormByClassName(FormClassName: string): TForm;
3899: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3900:   having Tag property value, equaled to Tag parameter }
3901: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3902: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3903: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3904: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3905: function RBTag(Parent: TWinControl): Integer;
3906: { AppMinimized returns True, if Application is minimized }
3907: function AppMinimized: Boolean;
3908: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3909:   if Caption parameter = '', it replaced with Application.Title }
3910: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3911: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3912:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3913: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3914:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3915: { Delay stop program execution to MSec msec }
3916: procedure Delay(MSec: Longword);
3917: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3918: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3919: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3920: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3921: function PanelBorder(Panel: TCustomPanel): Integer;
3922: function Pixels(Control: TControl; APixels: Integer): Integer;
3923: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3924: procedure Error(const Msg: string);
3925: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
```

```
3926:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3927:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3928: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3929:   const HideSelColor: Boolean): string;
3930: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3931:   const HideSelColor: Boolean): Integer;
3932: function ItemHtPlain(const Text: string): string;
3933: { ClearList - clears list of TObject }
3934: procedure ClearList(List: TList);
3935: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3936: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3937: { RTTI support }
3938: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3939: function GetPropStr(Obj: TObject; const PropName: string): string;
3940: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3941: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3942: procedure PrepareIniSection(SS: TStrings);
3943: { following functions are not documented because they are don't work properly, so don't use them }
3944: {$IFDEF COMPILER2}
3945: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3946: {$ENDIF}
3947:
3948: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3949: begin
3950:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3951:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3952:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
     Accept:Bool;Sorted:Bool;
3953:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3954:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3955:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3956:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3957:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3958: end;
3959:
3960: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3961: begin
3962:  Const('MaxInitStrNum','LongInt'( 9);
3963: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
     : array of AnsiString; MaxSplit : Integer) : Integer
3964: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
     string; MaxSplit : Integer) : Integer
3965: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
     QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3966: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3967: Function JvStrStrip( S : string) : string
3968: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3969: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3970: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3971: Function StrEatWhiteSpace( const S : string) : string
3972: Function HexToAscii( const S : AnsiString) : AnsiString
3973: Function AsciiToHex( const S : AnsiString) : AnsiString
3974: Function StripQuotes( const S1 : AnsiString) : AnsiString
3975: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3976: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3977: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3978: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3979: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3980: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
3981: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
3982: Function JvValidIdentifier( S1 : String) : Boolean
3983: Function JvEndChar( X : AnsiChar) : Boolean
3984: Procedure JvGetToken( S1, S2 : PAnsiChar)
3985: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
3986: Function IsKeyword( S1 : PAnsiChar) : Boolean
3987: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
3988: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
3989: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
3990: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
3991: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
3992: Function GetTokenCount : Integer
3993: Procedure ResetTokenCount
3994: end;
3995:
3996: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3997: begin
3998:   SIRegister_TJvQueryParamsDialog(CL);
3999:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
4000: end;
4001:
4002: ******************************** JvStrUtil /  JvStrUtils;****************************
4003: function FindNotBlankCharPos(const S: string): Integer;
4004: function AnsiChangeCase(const S: string): string;
4005: function GetWordOnPos(const S: string; const P: Integer): string;
4006: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4007: function Cmp(const S1, S2: string): Boolean;
4008: { Spaces returns string consists on N space chars }
4009: function Spaces(const N: Integer): string;
4010: { HasChar returns True, if char, Ch, contains in string, S }
```

```
4011: function HasChar(const Ch: Char; const S: string): Boolean;
4012: function HasAnyChar(const Chars: string; const S: string): Boolean;
4013: { SubStr returns substring from string, S, separated with Separator string}
4014: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
4015: { SubStrEnd same to previous function but Index numerated from the end of string }
4016: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4017: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
4018: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
4019: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
4020: { GetXYByPos is same to previous function, but returns X position in line too}
4021: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4022: { AddSlash returns string with added slash char to Dir parameter, if needed }
4023: function AddSlash2(const Dir: TFileName): string;
4024: { AddPath returns FileName with Path, if FileName not contain any path }
4025: function AddPath(const FileName, Path: TFileName): TFileName;
4026: { ExePath returns ExtractFilePath(ParamStr(0)) }
4027: function ExePath: TFileName;
4028: function LoadTextFile(const FileName: TFileName): string;
4029: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4030: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
4031: function ConcatSep(const S, S2, Separator: string): string;
4032: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4033: function FileEquMask(FileName, Mask: TFileName): Boolean;
4034: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4035:   Masks must be separated with comma (';') }
4036: function FileEquMasks(FileName, Masks: TFileName): Boolean;
4037: function StringEndsWith(const Str, SubStr: string): Boolean;
4038: function ExtractFilePath2(const FileName: string): string;
4039: function StrToOem(const AnsiStr: string): string;
4040: { StrToOem translates a string from the Windows character set into the OEM character set. }
4041: function OemToAnsiStr(const OemStr: string): string;
4042: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4043: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4044: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4045: function ReplaceStr(const S, Srch, Replace: string): string;
4046: { Returns string with every occurrence of Srch string replaced with Replace string. }
4047: function DelSpace(const S: string): string;
4048: { DelSpace return a string with all white spaces removed. }
4049: function DelChars(const S: string; Chr: Char): string;
4050: { DelChars return a string with all Chr characters removed. }
4051: function DelBSpace(const S: string): string;
4052: { DelBSpace trims leading spaces from the given string. }
4053: function DelESpace(const S: string): string;
4054: { DelESpace trims trailing spaces from the given string. }
4055: function DelRSpace(const S: string): string;
4056: { DelRSpace trims leading and trailing spaces from the given string. }
4057: function DelSpace1(const S: string): string;
4058: { DelSpace1 return a string with all non-single white spaces removed. }
4059: function Tab2Space(const S: string; Numb: Byte): string;
4060: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4061: function NPos(const C: string; S: string; N: Integer): Integer;
4062: { NPos searches for a N-th position of substring C in a given string. }
4063: function MakeStr(C: Char; N: Integer): string;
4064: function MS(C: Char; N: Integer): string;
4065: { MakeStr return a string of length N filled with character C. }
4066: function AddChar(C: Char; const S: string; N: Integer): string;
4067: { AddChar return a string left-padded to length N with characters C. }
4068: function AddCharR(C: Char; const S: string; N: Integer): string;
4069: { AddCharR return a string right-padded to length N with characters C. }
4070: function LeftStr(const S: string; N: Integer): string;
4071: { LeftStr return a string right-padded to length N with blanks. }
4072: function RightStr(const S: string; N: Integer): string;
4073: { RightStr return a string left-padded to length N with blanks. }
4074: function CenterStr(const S: string; Len: Integer): string;
4075: { CenterStr centers the characters in the string based upon the Len specified. }
4076: function CompStr(const S1, S2: string): Integer;
4077: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4078: function CompText(const S1, S2: string): Integer;
4079: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4080: function Copy2Symb(const S: string; Symb: Char): string;
4081: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4082: function Copy2SymbDel(var S: string; Symb: Char): string;
4083: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4084: function Copy2Space(const S: string): string;
4085: { Copy2Symb returns a substring of a string S from begining to first white space. }
4086: function Copy2SpaceDel(var S: string): string;
4087: { Copy2SpaceDel returns a substring of a string S from begining to first
4088:   white space and removes this substring from S. }
4089: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4090: { Returns string, with the first letter of each word in uppercase,
4091:   all other letters in lowercase. Words are delimited by WordDelims. }
4092: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4093: { WordCount given a set of word delimiters, returns number of words in S. }
4094: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4095: { Given a set of word delimiters, returns start position of N'th word in S. }
4096: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4097: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4098: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4099: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
```

```
4100:   delimiters, return the N'th word in S. }
4101: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4102: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4103: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4104: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4105: function QuotedString(const S: string; Quote: Char): string;
4106: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4107: function ExtractQuotedString(const S: string; Quote: Char): string;
4108: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4109:    and reduces pairs of Quote characters within the quoted string to a single character. }
4110: function FindPart(const HelpWilds, InputStr: string): Integer;
4111: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4112: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4113: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4114: function XorString(const Key, Src: ShortString): ShortString;
4115: function XorEncode(const Key, Source: string): string;
4116: function XorDecode(const Key, Source: string): string;
4117: { ** Command line routines ** }
4118: {$IFNDEF COMPILER4_UP}
4119: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4120: {$ENDIF}
4121: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4122: { ** Numeric string handling routines ** }
4123: function Numb2USA(const S: string): string;
4124: { Numb2USA converts numeric string S to USA-format. }
4125: function Dec2Hex(N: Longint; A: Byte): string;
4126: function D2H(N: Longint; A: Byte): string;
4127: { Dec2Hex converts the given value to a hexadecimal string representation
4128:   with the minimum number of digits (A) specified. }
4129: function Hex2Dec(const S: string): Longint;
4130: function H2D(const S: string): Longint;
4131: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4132: function Dec2Numb(N: Longint; A, B: Byte): string;
4133: { Dec2Numb converts the given value to a string representation with the
4134:   base equal to B and with the minimum number of digits (A) specified. }
4135: function Numb2Dec(S: string; B: Byte): Longint;
4136: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4137: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4138: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4139: function IntToRoman(Value: Longint): string;
4140: { IntToRoman converts the given value to a roman numeric string representation. }
4141: function RomanToInt(const S: string): Longint;
4142: { RomanToInt converts the given string to an integer value. If the string
4143:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4144: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4145: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4146: ********************************* JvFileUtil;*********************************
4147: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4148: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4149: procedure MoveFile(const FileName, DestName: TFileName);
4150: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4151: {$IFDEF COMPILER4_UP}
4152: function GetFileSize(const FileName: string): Int64;
4153: {$ELSE}
4154: function GetFileSize(const FileName: string): Longint;
4155: {$ENDIF}
4156: function FileDateTime(const FileName: string): TDateTime;
4157: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4158: function DeleteFiles(const FileMask: string): Boolean;
4159: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4160: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4161: function NormalDir(const DirName: string): string;
4162: function RemoveBackSlash(const DirName: string): string;
4163: function ValidFileName(const FileName: string): Boolean;
4164: function DirExists(Name: string): Boolean;
4165: procedure ForceDirectories(Dir: string);
4166: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4167:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4168: {$IFDEF COMPILER4_UP}
4169: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4170: {$ENDIF}
4171: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4172: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4173: {$IFDEF COMPILER4_UP}
4174: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4175: {$ENDIF}
4176: function GetTempDir: string;
4177: function GetWindowsDir: string;
4178: function GetSystemDir: string;
4179: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4180: {$IFDEF WIN32}
4181: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4182: function ShortToLongFileName(const ShortName: string): string;
4183: function ShortToLongPath(const ShortName: string): string;
4184: function LongToShortFileName(const LongName: string): string;
4185: function LongToShortPath(const LongName: string): string;
4186: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4187: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
```

```
4188: {$ENDIF WIN32}
4189: {$IFNDEF COMPILER3_UP}
4190: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4191: {$ENDIF}
4192: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4193: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4194: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4195: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4196:
4197: //*************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4198:  Procedure VariantClear( var V : Variant)
4199:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4200:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4201:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4202:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4203:  Procedure VariantSub( const src : Variant; var dst : Variant)
4204:  Procedure VariantMul( const src : Variant; var dst : Variant)
4205:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4206:  Procedure VariantMod( const src : Variant; var dst : Variant)
4207:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4208:  Procedure VariantOr( const src : Variant; var dst : Variant)
4209:  Procedure VariantXor( const src : Variant; var dst : Variant)
4210:  Procedure VariantShl( const src : Variant; var dst : Variant)
4211:  Procedure VariantShr( const src : Variant; var dst : Variant)
4212:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4213:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4214:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4215:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4216:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4217:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4218:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4219:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4220:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4221:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4222:  Function VariantNot( const V1 : Variant) : Variant
4223:  Function VariantNeg( const V1 : Variant) : Variant
4224:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4225:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4226:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4227:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4228:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4229:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4230:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4231:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4232:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4233:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4234: end;
4235:
4236: ************************************unit uPSI_JvgUtils;***********************************
4237: function IsEven(I: Integer): Boolean;
4238: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4239: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4240: procedure SwapInt(var I1, I2: Integer);
4241: function Spaces(Count: Integer): string;
4242: function DupStr(const Str: string; Count: Integer): string;
4243: function DupChar(C: Char; Count: Integer): string;
4244: procedure Msg(const AMsg: string);
4245: function RectW(R: TRect): Integer;
4246: function RectH(R: TRect): Integer;
4247: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4248: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4249: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4250: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4251:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4252: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4253: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4254:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4255: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4256: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4257:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4258: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4259: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4260: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4261:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4262:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4263:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4264: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4265:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4266:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4267:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4268: procedure BringParentWindowToTop(Wnd: TWinControl);
4269: function GetParentForm(Control: TControl): TForm;
4270: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4271: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4272: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4273: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4274: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4275: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
```

```
4276: function CalcMathString(AExpression: string): Single;
4277: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4278: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4279: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4280: procedure TypeStringOnKeyboard(const S: string);
4281: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4282: procedure DrawTextExtAligned(Canvas:TCanvas;const
      Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4283: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4284: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4285: function ComponentToString(Component: TComponent): string;
4286: procedure StringToComponent(Component: TComponent; const Value: string);
4287: function PlayWaveResource(const ResName: string): Boolean;
4288: function UserName: string;
4289: function ComputerName: string;
4290: function CreateIniFileName: string;
4291: function ExpandString(const Str: string; Len: Integer): string;
4292: function Transliterate(const Str: string; RusToLat: Boolean): string;
4293: function IsSmallFonts: Boolean;
4294: function SystemColorDepth: Integer;
4295: function GetFileTypeJ(const FileName: string): TglFileType;
4296: Function GetFileType( hFile : THandle) : DWORD');
4297: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4298: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4299:
4300: { ************************************Utility routines of unit classes}
4301: function LineStart(Buffer, BufPos: PChar): PChar
4302: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4303:   'Strings: TStrings): Integer
4304:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4305:  Procedure RegisterClass( AClass : TPersistentClass)
4306:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4307:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4308:  Procedure UnRegisterClass( AClass : TPersistentClass)
4309:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4310:  Procedure UnRegisterModuleClasses( Module : HMODULE)
4311:  Function FindGlobalComponent( const Name : string) : TComponent
4312:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4313:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4314:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4315:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4316:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4317:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4318:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4319:  Procedure GlobalFixupReferences
4320:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4321:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4322:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4323:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4324:  Procedure RemoveFixups( Instance : TPersistent)
4325:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4326:  Procedure BeginGlobalLoading
4327:  Procedure NotifyGlobalLoading
4328:  Procedure EndGlobalLoading
4329:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4330:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4331:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4332:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4333:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
4334: // Function AllocateHWnd( Method : TWndMethod) : HWND
4335:  Procedure DeallocateHWnd( Wnd : HWND)
4336:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4337: ************************************unit uPSI_SqlTimSt and DB;************************************
4338: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4339: Function VarSQLTimeStampCreate3: Variant;
4340: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4341: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4342: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4343: Function VarSQLTimeStamp : TVarType
4344: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4345: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4346: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4347: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4348: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4349: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4350: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4351: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4352: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4353: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4354: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4355: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4356: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4357:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4358: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4359: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4360: Procedure DisposeMem( var Buffer, Size : Integer)
4361: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4362: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4363: Function VarTypeToDataType( VarType : Integer) : TFieldType
```

```
4364: ********************************************unit JvStrings;*************************************
4365: {template functions}
4366: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4367: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4368: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4369: function RemoveMasterBlocks(const SourceStr: string): string;
4370: function RemoveFields(const SourceStr: string): string;
4371: {http functions}
4372: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4373: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4374: {set functions}
4375: procedure SplitSet(AText: string; AList: TStringList);
4376: function JoinSet(AList: TStringList): string;
4377: function FirstOfSet(const AText: string): string;
4378: function LastOfSet(const AText: string): string;
4379: function CountOfSet(const AText: string): Integer;
4380: function SetRotateRight(const AText: string): string;
4381: function SetRotateLeft(const AText: string): string;
4382: function SetPick(const AText: string; AIndex: Integer): string;
4383: function SetSort(const AText: string): string;
4384: function SetUnion(const Set1, Set2: string): string;
4385: function SetIntersect(const Set1, Set2: string): string;
4386: function SetExclude(const Set1, Set2: string): string;
4387: {replace any <,> etc by &lt; &gt;}
4388: function XMLSafe(const AText: string): string;
4389: {simple hash, Result can be used in Encrypt}
4390: function Hash(const AText: string): Integer;
4391: { Base64 encode and decode a string }
4392: function B64Encode(const S: AnsiString): AnsiString;
4393: function B64Decode(const S: AnsiString): AnsiString;
4394: {Basic encryption from a Borland Example}
4395: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4396: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4397: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4398: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4399: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4400: procedure CSVToTags(Src, Dst: TStringList);
4401: // converts a csv list to a tagged string list
4402: procedure TagsToCSV(Src, Dst: TStringList);
4403: // converts a tagged string list to a csv list
4404: // only fieldnames from the first record are scanned ib the other records
4405: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4406: {selects akey=avalue from Src and returns recordset in Dst}
4407: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4408: {filters Src for akey=avalue}
4409: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4410: {orders a tagged Src list by akey}
4411: function PosStr(const FindString, SourceString: string;
4412:   StartPos: Integer = 1): Integer;
4413: { PosStr searches the first occurrence of a substring FindString in a string
4414:   given by SourceString with case sensitivity (upper and lower case characters
4415:   are differed). This function returns the index value of the first character
4416:   of a specified substring from which it occurs in a given string starting with
4417:   StartPos character index. If a specified substring is not found Q_PosStr
4418:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4419: function PosStrLast(const FindString, SourceString: string): Integer;
4420: {finds the last occurance}
4421: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4422: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4423: { PosText searches the first occurrence of a substring FindString in a string
4424:   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
4425: function PosTextLast(const FindString, SourceString: string): Integer;
4426: {finds the last occurance}
4427: function NameValuesToXML(const AText: string): string;
4428: {$IFDEF MSWINDOWS}
4429: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4430: {$ENDIF MSWINDOWS}
4431: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4432: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4433: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4434: procedure SaveString(const AFile, AText: string);
4435: Procedure SaveStringasFile( const AFile, AText : string)
4436: function LoadStringJ(const AFile: string): string;
4437: Function LoadStringofFile( const AFile : string) : string
4438: Procedure SaveStringtoFile( const AFile, AText : string)
4439: Function LoadStringfromFile( const AFile : string) : string
4440: function HexToColor(const AText: string): TColor;
4441: function UppercaseHTMLTags(const AText: string): string;
4442: function LowercaseHTMLTags(const AText: string): string;
4443: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4444: function RelativePath(const ASrc, ADst: string): string;
4445: function GetToken(var Start: Integer; const SourceText: string): string;
4446: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4447: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4448: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4449: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4450: // parses the beginning of an attribute: space + alpha character
```

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

```
4539: { SubStrEnd same to previous function but Index numerated from the end of string }
4540: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4541: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4542: function SubWord(P: PChar; var P2: PChar): string;
4543: function CurrencyByWord(Value: Currency): string;
4544: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4545: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4546: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4547: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4548: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4549: { ReplaceString searches for all substrings, OldPattern,
4550:   in a string, S, and replaces them with NewPattern }
4551: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4552: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
     WideString;StartIndex:Integer=1):WideString;
4553: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4554: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4555: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4556: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4557:
4558: { Next 4 function for russian chars transliterating.
4559:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4560: procedure Dos2Win(var S: AnsiString);
4561: procedure Win2Dos(var S: AnsiString);
4562: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4563: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4564: function Win2Koi(const S: AnsiString): AnsiString;
4565: { FillString fills the string Buffer with Count Chars }
4566: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4567: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4568: { MoveString copies Count Chars from Source to Dest }
4569: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
     inline; {$ENDIF SUPPORTS_INLINE} overload;
4570: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4571:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4572: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4573: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4574: { MoveWideChar copies Count WideChars from Source to Dest }
4575: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
     SUPPORTS_INLINE}
4576: { FillNativeChar fills Buffer with Count NativeChars }
4577: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4578: { MoveWideChar copies Count WideChars from Source to Dest }
4579: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4580: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4581: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4582: { Spaces returns string consists on N space chars }
4583: function Spaces(const N: Integer): string;
4584: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4585: function AddSpaces(const S: string; const N: Integer): string;
4586: function SpacesW(const N: Integer): WideString;
4587: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4588: { function LastDateRUS for russian users only }
4589: { returns date relative to current date: 'ääà äíÿ íàçàä' }
4590: function LastDateRUS(const Dat: TDateTime): string;
4591: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4592: function CurrencyToStr(const Cur: Currency): string;
4593: { HasChar returns True, if Char, Ch, contains in string, S }
4594: function HasChar(const Ch: Char; const S: string): Boolean;
4595: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4596: function HasAnyChar(const Chars: string; const S: string): Boolean;
4597: {$IFNDEF COMPILER12_UP}
4598: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4599: {$ENDIF ~COMPILER12_UP}
4600: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
     SUPPORTS_INLINE}
4601: function CountOfChar(const Ch: Char; const S: string): Integer;
4602: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4603: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4604: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4605: function StrPosW(S, SubStr: PWideChar): PWideChar;
4606: function StrLenW(S: PWideChar): Integer;
4607: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4608: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
     SUPPORTS_INLINE}
4609: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4610: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4611: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4612:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4613: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4614: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4615: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4616: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4617: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
```

```
4618: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4619: Function ScreenPixelFormat : TPixelFormat
4620: Function ScreenColorCount : Integer
4621: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4622: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4623: //  SIRegister_TJvGradient(CL);
4624:
4625: {************************************ files routines}
4626: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4627: const
4628:   {$IFDEF MSWINDOWS}
4629:   DefaultCaseSensitivity = False;
4630:   {$ENDIF MSWINDOWS}
4631:   {$IFDEF UNIX}
4632:   DefaultCaseSensitivity = True;
4633:   {$ENDIF UNIX}
4634: { GenTempFileName returns temporary file name on
4635:   drive, there FileName is placed }
4636: function GenTempFileName(FileName: string): string;
4637: { GenTempFileNameExt same to previous function, but
4638:   returning filename has given extension, FileExt }
4639: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4640: { ClearDir clears folder Dir }
4641: function ClearDir(const Dir: string): Boolean;
4642: { DeleteDir clears and than delete folder Dir }
4643: function DeleteDir(const Dir: string): Boolean;
4644: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4645: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4646: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4647:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4648: function FileEquMasks(FileName, Masks: TFileName;
4649:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4650: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4651: {$IFDEF MSWINDOWS}
4652: { LZFileExpand expand file, FileSource,
4653:   into FileDest. Given file must be compressed, using MS Compress program }
4654: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4655: {$ENDIF MSWINDOWS}
4656: { FileGetInfo fills SearchRec record for specified file attributes}
4657: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4658: { HasSubFolder returns True, if folder APath contains other folders }
4659: function HasSubFolder(APath: TFileName): Boolean;
4660: { IsEmptyFolder returns True, if there are no files or
4661:   folders in given folder, APath}
4662: function IsEmptyFolder(APath: TFileName): Boolean;
4663: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4664: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4665: { AddPath returns FileName with Path, if FileName not contain any path }
4666: function AddPath(const FileName, Path: TFileName): TFileName;
4667: function AddPaths(const PathList, Path: string): string;
4668: function ParentPath(const Path: TFileName): TFileName;
4669: function FindInPath(const FileName, PathList: string): TFileName;
4670: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4671: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4672: { HasParam returns True, if program running with specified parameter, Param }
4673: function HasParam(const Param: string): Boolean;
4674: function HasSwitch(const Param: string): Boolean;
4675: function Switch(const Param: string): string;
4676: { ExePath returns ExtractFilePath(ParamStr(0)) }
4677: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4678: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4679: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4680: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4681: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4682: {**** Graphic routines }
4683: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4684: function IsTTFontSelected(const DC: HDC): Boolean;
4685: function KeyPressed(VK: Integer): Boolean;
4686: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4687: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4688: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4689: {**** Color routines }
4690: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4691: function RGBToBGR(Value: Cardinal): Cardinal;
4692: //function ColorToPrettyName(Value: TColor): string;
4693: //function PrettyNameToColor(const Value: string): TColor;
4694: {**** other routines }
4695: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4696: function IntPower(Base, Exponent: Integer): Integer;
4697: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4698: function StrToBool(const S: string): Boolean;
4699: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4700: function VarToInt(V: Variant): Integer;
4701: function VarToFloat(V: Variant): Double;
4702: { following functions are not documented because they not work properly sometimes, so do not use them }
4703: // (rom) ReplaceStrings1, GetSubStr removed
4704: function GetLongFileName(const FileName: string): string;
4705: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4706: function GetParameter: string;
```

```
4707: function GetComputerID: string;
4708: function GetComputerName: string;
4709: {**** string routines }
4710: { ReplaceAllStrings searches for all substrings, Words,
4711:   in a string, S, and replaces them with Frases with the same Index. }
4712: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4713: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4714:   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 }
4715: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4716: { CountOfLines calculates the lines count in a string, S,
4717:   each line must be separated from another with CrLf sequence }
4718: function CountOfLines(const S: string): Integer;
4719: { DeleteLines deletes all lines from strings which in the words,  words.
4720:   The word of will be deleted from strings. }
4721: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4722: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4723:   Lines contained only spaces also deletes. }
4724: procedure DeleteEmptyLines(Ss: TStrings);
4725: { SQLAddWhere addes or modifies existing where-statement, where,
4726:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4727:   it must be started on the begining of any line }
4728: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4729: {**** files routines - }
4730: {$IFDEF MSWINDOWS}
4731: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4732:   Resource can be compressed using MS Compress program}
4733: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4734: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
      Boolean;
4735: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4736: {$ENDIF MSWINDOWS}
4737: { IniReadSection read section, Section, from ini-file,
4738:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4739:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4740: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4741: { LoadTextFile load text file, FileName, into string }
4742: function LoadTextFile(const FileName: TFileName): string;
4743: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4744: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4745: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4746: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4747: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4748: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4749: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4750: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4751: { RATextCalcHeight calculate needed height for
4752:   correct output, using RATextOut or RATextOutEx functions }
4753: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4754: { Cinema draws some visual effect }
4755: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4756: { Roughed fills rect with special 3D pattern }
4757: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4758: { 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 }
4759: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4760: { TextWidth calculate text with for writing using standard desktop font }
4761: function TextWidth(const AStr: string): Integer;
4762: { TextHeight calculate text height for writing using standard desktop font }
4763: function TextHeight(const AStr: string): Integer;
4764: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4765: procedure Error(const Msg: string);
4766: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4767:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4768: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4769: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4770:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4771: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4772:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4773: function ItemHtPlain(const Text: string): string;
4774: { ClearList - clears list of TObject }
4775: procedure ClearList(List: TList);
4776: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4777: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4778: { RTTI support }
4779: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4780: function GetPropStr(Obj: TObject; const PropName: string): string;
4781: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4782: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4783: procedure PrepareIniSection(Ss: TStrings);
4784: { following functions are not documented because they are don't work properly, so don't use them }
4785: // (rom) from JvBandWindows to make it obsolete
4786: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4787: // (rom) from JvBandUtils to make it obsolete
4788: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4789: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4790: function CreateIconFromClipboard: TIcon;
4791: { begin JvIconClipboardUtils } { Icon clipboard routines }
4792: function CF_ICON: Word;
```

```
4793: procedure AssignClipboardIcon(Icon: TIcon);
4794: { Real-size icons support routines (32-bit only) }
4795: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4796: function CreateRealSizeIcon(Icon: TIcon): HICON;
4797: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4798: {end JvIconClipboardUtils }
4799: function CreateScreenCompatibleDC: HDC;
4800: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF}
4801: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4802: { begin JvRLE } // (rom) changed API for inclusion in JCL
4803: procedure RleCompressTo(InStream, OutStream: TStream);
4804: procedure RleDecompressTo(InStream, OutStream: TStream);
4805: procedure RleCompress(Stream: TStream);
4806: procedure RleDecompress(Stream: TStream);
4807: { end JvRLE } { begin JvDateUtil }
4808: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4809: function IsLeapYear(AYear: Integer): Boolean;
4810: function DaysInAMonth(const AYear, AMonth: Word): Word;
4811: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4812: function FirstDayOfPrevMonth: TDateTime;
4813: function LastDayOfPrevMonth: TDateTime;
4814: function FirstDayOfNextMonth: TDateTime;
4815: function ExtractDay(ADate: TDateTime): Word;
4816: function ExtractMonth(ADate: TDateTime): Word;
4817: function ExtractYear(ADate: TDateTime): Word;
4818: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4819: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4820: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4821: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4822: function ValidDate(ADate: TDateTime): Boolean;
4823: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4824: function MonthsBetween(Date1, Date2: TDateTime): Double;
4825: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4826: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4827: function DaysBetween(Date1, Date2: TDateTime): Longint;
4828: { The same as previous but if Date2 < Date1 result = 0 }
4829: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4830: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4831: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4832: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4833: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4834: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4835: { String to date conversions }
4836: function GetDateOrder(const DateFormat: string): TDateOrder;
4837: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4838: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4839: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4840: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4841: //function DefDateFormat(AFourDigitYear: Boolean): string;
4842: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4843: function FormatLongDate(Value: TDateTime): string;
4844: function FormatLongDateTime(Value: TDateTime): string;
4845: { end JvDateUtil }
4846: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4847: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4848: { begin JvStrUtils } { ** Common string handling routines ** }
4849: {$IFDEF UNIX}
4850: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4851:   const ToCode, FromCode: AnsiString): Boolean;
4852: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4853: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4854: function OemStrToAnsi(const S: AnsiString): AnsiString;
4855: function AnsiStrToOem(const S: AnsiString): AnsiString;
4856: {$ENDIF UNIX}
4857: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4858: { StrToOem translates a string from the Windows character set into the OEM character set. }
4859: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4860: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4861: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4862: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4863: function ReplaceStr(const S, Srch, Replace: string): string;
4864: { Returns string with every occurrence of Srch string replaced with Replace string. }
4865: function DelSpace(const S: string): string;
4866: { DelSpace return a string with all white spaces removed. }
4867: function DelChars(const S: string; Chr: Char): string;
4868: { DelChars return a string with all Chr characters removed. }
4869: function DelBSpace(const S: string): string;
4870: { DelBSpace trims leading spaces from the given string. }
4871: function DelESpace(const S: string): string;
4872: { DelESpace trims trailing spaces from the given string. }
4873: function DelRSpace(const S: string): string;
4874: { DelRSpace trims leading and trailing spaces from the given string. }
4875: function DelSpace1(const S: string): string;
4876: { DelSpace1 return a string with all non-single white spaces removed. }
4877: function Tab2Space(const S: string; Numb: Byte): string;
4878: { Tab2Space converts any tabulation character in the given string to the
4879:   Numb spaces characters. }
4880: function NPos(const C: string; S: string; N: Integer): Integer;
```

```
4881: { NPos searches for a N-th position of substring C in a given string. }
4882: function MakeStr(C: Char; N: Integer): string; overload;
4883: {$IFNDEF COMPILER12_UP}
4884: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4885: {$ENDIF !COMPILER12_UP}
4886: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4887: { MakeStr return a string of length N filled with character C. }
4888: function AddChar(C: Char; const S: string; N: Integer): string;
4889: { AddChar return a string left-padded to length N with characters C. }
4890: function AddCharR(C: Char; const S: string; N: Integer): string;
4891: { AddCharR return a string right-padded to length N with characters C. }
4892: function LeftStr(const S: string; N: Integer): string;
4893: { LeftStr return a string right-padded to length N with blanks. }
4894: function RightStr(const S: string; N: Integer): string;
4895: { RightStr return a string left-padded to length N with blanks. }
4896: function CenterStr(const S: string; Len: Integer): string;
4897: { CenterStr centers the characters in the string based upon the Len specified. }
4898: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4899: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4900:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4901: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4902: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4903: function Copy2Symb(const S: string; Symb: Char): string;
4904: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4905: function Copy2SymbDel(var S: string; Symb: Char): string;
4906: { Copy2SymbDel returns a substring of a string S from begining to first
4907:   character Symb and removes this substring from S. }
4908: function Copy2Space(const S: string): string;
4909: { Copy2Symb returns a substring of a string S from begining to first white space. }
4910: function Copy2SpaceDel(var S: string): string;
4911: { Copy2SpaceDel returns a substring of a string S from begining to first
4912:   white space and removes this substring from S. }
4913: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4914: { Returns string, with the first letter of each word in uppercase,
4915:   all other letters in lowercase. Words are delimited by WordDelims. }
4916: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4917: { WordCount given a set of word delimiters, returns number of words in S. }
4918: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4919: { Given a set of word delimiters, returns start position of N'th word in S. }
4920: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4921: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4922: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4923: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4924:   delimiters, return the N'th word in S. }
4925: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4926: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4927:   that started from position Pos. }
4928: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4929: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4930: function QuotedString(const S: string; Quote: Char): string;
4931: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4932: function ExtractQuotedString(const S: string; Quote: Char): string;
4933: { ExtractQuotedString removes the Quote characters from the beginning and
4934:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4935: function FindPart(const HelpWilds, InputStr: string): Integer;
4936: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4937: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4938: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4939: function XorString(const Key, Src: ShortString): ShortString;
4940: function XorEncode(const Key, Source: string): string;
4941: function XorDecode(const Key, Source: string): string;
4942: { ** Command line routines ** }
4943: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4944: { ** Numeric string handling routines ** }
4945: function Numb2USA(const S: string): string;
4946: { Numb2USA converts numeric string S to USA-format. }
4947: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4948: { Dec2Hex converts the given value to a hexadecimal string representation
4949:   with the minimum number of digits (A) specified. }
4950: function Hex2Dec(const S: string): Longint;
4951: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4952: function Dec2Numb(N: Int64; A, B: Byte): string;
4953: { Dec2Numb converts the given value to a string representation with the
4954:   base equal to B and with the minimum number of digits (A) specified. }
4955: function Numb2Dec(S: string; B: Byte): Int64;
4956: { Numb2Dec converts the given B-based numeric string to the corresponding
4957:   integer value. }
4958: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4959: { IntToBin converts the given value to a binary string representation
4960:   with the minimum number of digits specified. }
4961: function IntToRoman(Value: Longint): string;
4962: { IntToRoman converts the given value to a roman numeric string representation. }
4963: function RomanToInt(const S: string): Longint;
4964: { RomanToInt converts the given string to an integer value. If the string
4965:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4966: function FindNotBlankCharPos(const S: string): Integer;
4967: function FindNotBlankCharPosW(const S: WideString): Integer;
4968: function AnsiChangeCase(const S: string): string;
4969: function WideChangeCase(const S: string): string;
```

```
4970: function StartsText(const SubStr, S: string): Boolean;
4971: function EndsText(const SubStr, S: string): Boolean;
4972: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4973: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4974: {end JvStrUtils}
4975: {$IFDEF UNIX}
4976: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4977: {$ENDIF UNIX}
4978: { begin JvFileUtil }
4979: function FileDateTime(const FileName: string): TDateTime;
4980: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4981: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4982: function NormalDir(const DirName: string): string;
4983: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4984: function ValidFileName(const FileName: string): Boolean;
4985: {$IFDEF MSWINDOWS}
4986: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4987: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4988: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4989: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4990: {$ENDIF MSWINDOWS}
4991: function GetWindowsDir: string;
4992: function GetSystemDir: string;
4993: function ShortToLongFileName(const ShortName: string): string;
4994: function LongToShortFileName(const LongName: string): string;
4995: function ShortToLongPath(const ShortName: string): string;
4996: function LongToShortPath(const LongName: string): string;
4997: {$IFDEF MSWINDOWS}
4998: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4999: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
5000: {$ENDIF MSWINDOWS}
5001: { end JvFileUtil }
5002: // Works like PtInRect but includes all edges in comparision
5003: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
5004: // Works like PtInRect but excludes all edges from comparision
5005: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
5006: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
5007: function IsFourDigitYear: Boolean;
5008: { moved from JvJVCLUTils }
5009: //Open an object with the shell (url or something like that)
5010: function OpenObject(const Value: string): Boolean; overload;
5011: function OpenObject(Value: PChar): Boolean; overload;
5012: {$IFDEF MSWINDOWS}
5013: //Raise the last Exception
5014: procedure RaiseLastWin32; overload;
5015: procedure RaiseLastWin32(const Text: string); overload;
5016: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
5017: function GetFileVersion(const AFileName: string): Cardinal;
5018: {$EXTERNALSYM GetFileVersion}
5019: //Get version of Shell.dll
5020: function GetShellVersion: Cardinal;
5021: {$EXTERNALSYM GetShellVersion}
5022: // CD functions  on HW
5023: procedure OpenCdDrive;
5024: procedure CloseCdDrive;
5025: // returns True if Drive is accessible
5026: function DiskInDrive(Drive: Char): Boolean;
5027: {$ENDIF MSWINDOWS}
5028: //Same as linux function ;)
5029: procedure PError(const Text: string);
5030: // execute a program without waiting
5031: procedure Exec(const FileName, Parameters, Directory: string);
5032: // execute a program and wait for it to finish
5033: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
5034: // returns True if this is the first instance of the program that is running
5035: function FirstInstance(const ATitle: string): Boolean;
5036: // restores a window based on it's classname and Caption. Either can be left empty
5037: // to widen the search
5038: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
5039: // manipulate the traybar and start button
5040: procedure HideTraybar;
5041: procedure ShowTraybar;
5042: procedure ShowStartButton(Visible: Boolean = True);
5043: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5044: procedure MonitorOn;
5045: procedure MonitorOff;
5046: procedure LowPower;
5047: // send a key to the window named AppName
5048: function SendKey(const AppName: string; Key: Char): Boolean;
5049: {$IFDEF MSWINDOWS}
5050: // returns a list of all win currently visible, the Objects property is filled with their window handle
5051: procedure GetVisibleWindows(List: TStrings);
5052: Function GetVisibleWindowsF( List : TStrings):TStrings');
5053: // associates an extension to a specific program
5054: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5055: procedure AddToRecentDocs(const FileName: string);
5056: function GetRecentDocs: TStringList;
```

```
5057: {$ENDIF MSWINDOWS}
5058: function CharIsMoney(const Ch: Char): Boolean;
5059: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5060: function IntToExtended(I: Integer): Extended;
5061: { GetChangedText works out the new text given the current cursor pos & the key pressed
5062:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5063: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5064: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5065: //function StrIsInteger(const S: string): Boolean;
5066: function StrIsFloatMoney(const Ps: string): Boolean;
5067: function StrIsDateTime(const Ps: string): Boolean;
5068: function PreformatDateString(Ps: string): string;
5069: function BooleanToInteger(const B: Boolean): Integer;
5070: function StringToBoolean(const Ps: string): Boolean;
5071: function SafeStrToDateTime(const Ps: string): TDateTime;
5072: function SafeStrToDate(const Ps: string): TDateTime;
5073: function SafeStrToTime(const Ps: string): TDateTime;
5074: function StrDelete(const psSub, psMain: string): string;
5075:   { returns the fractional value of pcValue}
5076: function TimeOnly(pcValue: TDateTime): TTime;
5077: { returns the integral value of pcValue }
5078: function DateOnly(pcValue: TDateTime): TDate;
5079: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5080: const { TDateTime value used to signify Null value}
5081:   NullEquivalentDate: TDateTime = 0.0;
5082: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5083: // Replacement for Win32Check to avoid platform specific warnings in D6
5084: function OSCheck(RetVal: Boolean): Boolean;
5085: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5086:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5087:   not be forced to use FileCtrl unnecessarily }
5088: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5089: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5090: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5091: function MinimizeString(const S: string; const MaxLen: Integer): string;
5092: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
      SW_SHOWDEFAULT);
5093: { 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.}
5094: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5095: {$ENDIF MSWINDOWS}
5096: procedure ResourceNotFound(ResID: PChar);
5097: function EmptyRect: TRect;
5098: function RectWidth(R: TRect): Integer;
5099: function RectHeight(R: TRect): Integer;
5100: function CompareRect(const R1, R2: TRect): Boolean;
5101: procedure RectNormalize(var R: TRect);
5102: function RectIsSquare(const R: TRect): Boolean;
5103: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5104: //If AMaxSize = -1 ,then auto calc Square's max size
5105: {$IFDEF MSWINDOWS}
5106: procedure FreeUnusedOle;
5107: function GetWindowsVersion: string;
5108: function LoadDLL(const LibName: string): THandle;
5109: function RegisterServer(const ModuleName: string): Boolean;
5110: function UnregisterServer(const ModuleName: string): Boolean;
5111: {$ENDIF MSWINDOWS}
5112: { String routines }
5113: function GetEnvVar(const VarName: string): string;
5114: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5115: function StringToPChar(var S: string): PChar;
5116: function StrPAlloc(const S: string): PChar;
5117: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5118: function DropT(const S: string): string;
5119: { Memory routines }
5120: function AllocMemo(Size: Longint): Pointer;
5121: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5122: procedure FreeMemo(var fpBlock: Pointer);
5123: function GetMemoSize(fpBlock: Pointer): Longint;
5124: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5125: { Manipulate huge pointers routines }
5126: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5127: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5128: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5129: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5130: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5131: function WindowClassName(Wnd: THandle): string;
5132: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5133: procedure ActivateWindow(Wnd: THandle);
5134: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5135: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5136: { SetWindowTop put window to top without recreating window }
5137: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5138: procedure CenterWindow(Wnd: THandle);
5139: function MakeVariant(const Values: array of Variant): Variant;
5140: { Convert dialog units to pixels and backwards }
5141: {$IFDEF MSWINDOWS}
5142: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
```

```
5143: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5144: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5145: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5146: {$ENDIF MSWINDOWS}
5147: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5148: {$IFDEF BCB}
5149: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5150: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5151: {$ELSE}
5152: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5153: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5154: {$ENDIF BCB}
5155: {$IFDEF MSWINDOWS}
5156: { BrowseForFolderNative displays Browse For Folder dialog }
5157: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5158: {$ENDIF MSWINDOWS}
5159: procedure AntiAlias(Clip: TBitmap);
5160: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5161: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5162:   ABitmap: TBitmap; const SourceRect: TRect);
5163: function IsTrueType(const FontName: string): Boolean;
5164: // Removes all non-numeric characters from AValue and returns the resulting string
5165: function TextToValText(const AValue: string): string;
5166: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5167: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5168: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5169: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5170: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5171:
5172: ****************************************************unit uPSI_JvTFUtils;
5173:  Function JExtractYear( ADate : TDateTime) : Word
5174:  Function JExtractMonth( ADate : TDateTime) : Word
5175:  Function JExtractDay( ADate : TDateTime) : Word
5176:  Function ExtractHours( ATime : TDateTime) : Word
5177:  Function ExtractMins( ATime : TDateTime) : Word
5178:  Function ExtractSecs( ATime : TDateTime) : Word
5179:  Function ExtractMSecs( ATime : TDateTime) : Word
5180:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5181:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5182:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5183:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5184:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5185:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5186:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5187:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5188:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5189:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5190:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5191:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5192:  Procedure EnsureMonth( Month : Word)
5193:  Procedure EnsureDOW( DOW : Word)
5194:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5195:  Function Lesser( N1, N2 : Integer) : Integer
5196:  Function Greater( N1, N2 : Integer) : Integer
5197:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5198:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5199:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5200:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5201:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5202:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5203:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
     AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5204:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
     HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5205:  Function JRectWidth( ARect : TRect) : Integer
5206:  Function JRectHeight( ARect : TRect) : Integer
5207:  Function JEmptyRect : TRect
5208:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5209:
5210: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5211: begin
5212:  Procedure HideTaskBarButton( hWindow : HWND)
5213:  Function msLoadStr( ID : Integer) : String
5214:  Function msFormat( fmt : String; params : array of const) : String
5215:  Function msFileExists( const FileName : String) : Boolean
5216:  Function msIntToStr( Int : Int64) : String
5217:  Function msStrPas( const Str : PChar) : String
5218:  Function msRenameFile( const OldName, NewName : String) : Boolean
5219:  Function CutFileName( s : String) : String
5220:  Function GetVersionInfo( var VersionString : String) : DWORD
5221:  Function FormatTime( t : Cardinal) : String
5222:  Function msCreateDir( const Dir : string) : Boolean
5223:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5224:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5225:  Function msStrLen( Str : PChar) : Integer
5226:  Function msDirectoryExists( const Directory : String) : Boolean
5227:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5228:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
```

```
5229:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5230:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5231:  Function GetTextFromFile( Filename : String) : string
5232:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5233:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5234:  Function msStrToInt( s : String) : Integer
5235:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
5236: end;
5237:
5238: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5239: begin
5240:   //TDynFloatArray', 'array of Extended
5241:   TDynLWordArray', 'array of LongWord
5242:   TDynLIntArray', 'array of LongInt
5243:   TDynFloatMatrix', 'array of TDynFloatArray
5244:   TDynLWordMatrix', 'array of TDynLWordArray
5245:   TDynLIntMatrix', 'array of TDynLIntArray
5246:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5247:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5248:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5249:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5250:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5251:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5252:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5253:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5254:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5255:  Function MNorm( const X : TDynFloatArray) : Extended
5256:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5257:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5258:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5259:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5260:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5261:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5262:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5263:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5264:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5265:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5266:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5267:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5268:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5269: end;
5270:
5271: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5272: begin
5273:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5274:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5275:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5276:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5277:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5278:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5279:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5280:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5281:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5282:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5283:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5284:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5285:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5286:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5287:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5288:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5289:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5290:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5291:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5292:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5293:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5294:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5295:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5296:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5297:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5298:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5299:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5300:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5301:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5302:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
5303:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5304:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5305:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5306:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5307:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5308:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5309:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5310:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5311:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5312:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5313:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5314:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5315:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5316:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5317:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
```

```
5318:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5319:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5320:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5321:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5322:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5323:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5324:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5325:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5326:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5327:   //LongWord', 'Cardinal
5328:   TBitList', 'Word
5329:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5330:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5331:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5332:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5333:  Function SameFloat( const X1, X2 : Extended) : Boolean
5334:  Function FloatIsZero( const X : Extended) : Boolean
5335:  Function FloatIsPositive( const X : Extended) : Boolean
5336:  Function FloatIsNegative( const X : Extended) : Boolean
5337:  Procedure IncLim( var B : Byte; const Limit : Byte)
5338:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5339:  Procedure IncLimW( var B : Word; const Limit : Word)
5340:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5341:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5342:  Procedure DecLim( var B : Byte; const Limit : Byte)
5343:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5344:  Procedure DecLimW( var B : Word; const Limit : Word)
5345:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5346:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5347:  Function MaxB( const B1, B2 : Byte) : Byte
5348:  Function MinB( const B1, B2 : Byte) : Byte
5349:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5350:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5351:  Function MaxW( const B1, B2 : Word) : Word
5352:  Function MinW( const B1, B2 : Word) : Word
5353:  Function esbMaxI( const B1, B2 : Integer) : Integer
5354:  Function esbMinI( const B1, B2 : Integer) : Integer
5355:  Function MaxL( const B1, B2 : LongInt) : LongInt
5356:  Function MinL( const B1, B2 : LongInt) : LongInt
5357:  Procedure SwapB( var B1, B2 : Byte)
5358:  Procedure SwapSI( var B1, B2 : ShortInt)
5359:  Procedure SwapW( var B1, B2 : Word)
5360:  Procedure SwapI( var B1, B2 : SmallInt)
5361:  Procedure SwapL( var B1, B2 : LongInt)
5362:  Procedure SwapI32( var B1, B2 : Integer)
5363:  Procedure SwapC( var B1, B2 : LongWord)
5364:  Procedure SwapInt64( var X, Y : Int64)
5365:  Function esbSign( const B : LongInt) : ShortInt
5366:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5367:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5368:  Function Max3Word( const X1, X2, X3 : Word) : Word
5369:  Function Min3Word( const X1, X2, X3 : Word) : Word
5370:  Function MaxBArray( const B : array of Byte) : Byte
5371:  Function MaxWArray( const B : array of Word) : Word
5372:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5373:  Function MaxIArray( const B : array of Integer) : Integer
5374:  Function MaxLArray( const B : array of LongInt) : LongInt
5375:  Function MinBArray( const B : array of Byte) : Byte
5376:  Function MinWArray( const B : array of Word) : Word
5377:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5378:  Function MinIArray( const B : array of Integer) : Integer
5379:  Function MinLArray( const B : array of LongInt) : LongInt
5380:  Function SumBArray( const B : array of Byte) : Byte
5381:  Function SumBArray2( const B : array of Byte) : Word
5382:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5383:  Function SumSIArray2( const B : array of ShortInt) : Integer
5384:  Function SumWArray( const B : array of Word) : Word
5385:  Function SumWArray2( const B : array of Word) : LongInt
5386:  Function SumIArray( const B : array of Integer) : Integer
5387:  Function SumLArray( const B : array of LongInt) : LongInt
5388:  Function SumLWArray( const B : array of LongWord) : LongWord
5389:  Function ESBDigits( const X : LongWord) : Byte
5390:  Function BitsHighest( const X : LongWord) : Integer
5391:  Function ESBBitsNeeded( const X : LongWord) : Integer
5392:  Function esbGCD( const X, Y : LongWord) : LongWord
5393:  Function esbLCM( const X, Y : LongInt) : Int64
5394:  //Function esbLCM( const X, Y : LongInt) : LongInt
5395:  Function RelativePrime( const X, Y : LongWord) : Boolean
5396:  Function Get87ControlWord : TBitList
5397:  Procedure Set87ControlWord( const CWord : TBitList)
5398:  Procedure SwapExt( var X, Y : Extended)
5399:  Procedure SwapDbl( var X, Y : Double)
5400:  Procedure SwapSing( var X, Y : Single)
5401:  Function esbSgn( const X : Extended) : ShortInt
5402:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5403:  Function ExtMod( const X, Y : Extended) : Extended
5404:  Function ExtRem( const X, Y : Extended) : Extended
5405:  Function CompMOD( const X, Y : Comp) : Comp
5406:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
```

```
5407:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5408:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5409:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5410:  Function MaxExt( const X, Y : Extended) : Extended
5411:  Function MinExt( const X, Y : Extended) : Extended
5412:  Function MaxEArray( const B : array of Extended) : Extended
5413:  Function MinEArray( const B : array of Extended) : Extended
5414:  Function MaxSArray( const B : array of Single) : Single
5415:  Function MinSArray( const B : array of Single) : Single
5416:  Function MaxCArray( const B : array of Comp) : Comp
5417:  Function MinCArray( const B : array of Comp) : Comp
5418:  Function SumSArray( const B : array of Single) : Single
5419:  Function SumEArray( const B : array of Extended) : Extended
5420:  Function SumSqEArray( const B : array of Extended) : Extended
5421:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5422:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5423:  Function SumCArray( const B : array of Comp) : Comp
5424:  Function FactorialX( A : LongWord) : Extended
5425:  Function PermutationX( N, R : LongWord) : Extended
5426:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5427:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5428:  Function esbGeometricMean( const X : array of Extended) : Extended
5429:  Function esbHarmonicMean( const X : array of Extended) : Extended
5430:  Function ESBMean( const X : array of Extended) : Extended
5431:  Function esbSampleVariance( const X : array of Extended) : Extended
5432:  Function esbPopulationVariance( const X : array of Extended) : Extended
5433:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5434:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5435:  Function GetMedian( const SortedX : array of Extended) : Extended
5436:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5437:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5438:  Function ESBMagnitude( const X : Extended) : Integer
5439:  Function ESBTan( Angle : Extended) : Extended
5440:  Function ESBCot( Angle : Extended) : Extended
5441:  Function ESBCosec( const Angle : Extended) : Extended
5442:  Function ESBSec( const Angle : Extended) : Extended
5443:  Function ESBArcTan( X, Y : Extended) : Extended
5444:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5445:  Function ESBArcCos( const X : Extended) : Extended
5446:  Function ESBArcSin( const X : Extended) : Extended
5447:  Function ESBArcSec( const X : Extended) : Extended
5448:  Function ESBArcCosec( const X : Extended) : Extended
5449:  Function ESBLog10( const X : Extended) : Extended
5450:  Function ESBLog2( const X : Extended) : Extended
5451:  Function ESBLogBase( const X, Base : Extended) : Extended
5452:  Function Pow2( const X : Extended) : Extended
5453:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5454:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5455:  Function XtoY( const X, Y : Extended) : Extended
5456:  Function esbTenToY( const Y : Extended) : Extended
5457:  Function esbTwoToY( const Y : Extended) : Extended
5458:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5459:  Function esbISqrt( const I : LongWord) : Longword
5460:  Function ILog2( const I : LongWord) : LongWord
5461:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5462:  Function ESBArCosh( X : Extended) : Extended
5463:  Function ESBArSinh( X : Extended) : Extended
5464:  Function ESBArTanh( X : Extended) : Extended
5465:  Function ESBCosh( X : Extended) : Extended
5466:  Function ESBSinh( X : Extended) : Extended
5467:  Function ESBTanh( X : Extended) : Extended
5468:  Function InverseGamma( const X : Extended) : Extended
5469:  Function esbGamma( const X : Extended) : Extended
5470:  Function esbLnGamma( const X : Extended) : Extended
5471:  Function esbBeta( const X, Y : Extended) : Extended
5472:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5473: end;
5474:
5475: ********************************Integer Huge Cardinal Utils*************************
5476: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5477: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5478: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5479: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5480: Function BitCount_8( Value : byte) : integer
5481: Function BitCount_16( Value : uint16) : integer
5482: Function BitCount_32( Value : uint32) : integer
5483: Function BitCount_64( Value : uint64) : integer
5484: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5485:  Procedure ( CountPrimalityTests : integer)
5486:  Function gcd( a, b : THugeCardinal) : THugeCardinal
5487:  Function lcm( a, b : THugeCardinal) : THugeCardinal
5488:  Function isCoPrime( a, b : THugeCardinal) : boolean
5489:  Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean
5490:  Function hasSmallFactor( p : THugeCardinal) : boolean
5491:  //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
      NumbersTested: integer) : boolean
5492:  Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5493:  Const('StandardExponent','LongInt'( 65537);
```

```
5494: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5495:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5496:
5497: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5498: begin
5499:   AddTypeS('TXRTLInteger', 'array of Integer
5500:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5501:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5502:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5503:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5504:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5505:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5506:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5507:   'BitsPerByte','LongInt'( 8);
5508:   BitsPerDigit','LongInt'( 32);
5509:   SignBitMask','LongWord( $80000000);
5510:  Function XRTLAdjustBits( const ABits : Integer) : Integer
5511:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5512:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5513:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5514:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5515:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5516:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5517:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5518:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5519:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5520:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5521:  Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5522:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5523:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5524:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5525:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5526:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5527:  Procedure XRTLZero( var AInteger : TXRTLInteger)
5528:  Procedure XRTLOne( var AInteger : TXRTLInteger)
5529:  Procedure XRTLMOne( var AInteger : TXRTLInteger)
5530:  Procedure XRTLTwo( var AInteger : TXRTLInteger)
5531:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5532:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5533:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5534:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5535:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5536:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5537:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5538:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5539:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5540:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5541:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5542:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5543:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5544:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5545:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5546:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5547:  Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger)
5548:  Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5549:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5550:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5551:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5552:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5553:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5554:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5555:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5556:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5557:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5558:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5559:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5560:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5561:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5562:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5563:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5564:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5565:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5566:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5567:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5568:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5569:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5570:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5571:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5572:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5573:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5574:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5575:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5576:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5577:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5578:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
```

```
5579:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5580:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5581:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5582:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5583:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5584: end;
5585:
5586: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5587: begin
5588:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5589:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5590:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5591:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5592:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5593:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5594:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5595:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5596:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5597:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5598:  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)
5599: end;
5600:
5601:
5602: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5603: begin
5604:  Function StrDec( S : String) : String
5605:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5606:  Function ReadNumFromEdit( Edit : TEdit) : Float
5607:  Procedure WriteNumToFile( var F : Text; X : Float)
5608: end;
5609:
5610: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5611: begin
5612:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5613:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5614:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5615:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5616:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5617:  Procedure TeX_SetGraphTitle( Title : String)
5618:  Procedure TeX_SetOxTitle( Title : String)
5619:  Procedure TeX_SetOyTitle( Title : String)
5620:  Procedure TeX_PlotOxAxis
5621:  Procedure TeX_PlotOyAxis
5622:  Procedure TeX_PlotGrid( Grid : TGrid)
5623:  Procedure TeX_WriteGraphTitle
5624:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5625:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5626:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5627:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5628:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5629:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5630:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5631:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5632:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5633:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5634:  Function Xcm( X : Float) : Float
5635:  Function Ycm( Y : Float) : Float
5636: end;
5637:
5638: *---------------------------------------------------------------------------*)
5639: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5640: begin
5641:   TConstArray', 'array of TVarRec
5642:  Function CopyVarRec( const Item : TVarRec) : TVarRec
5643:  Function CreateConstArray( const Elements : array of const) : TConstArray
5644:  Procedure FinalizeVarRec( var Item : TVarRec)
5645:  Procedure FinalizeConstArray( var Arr : TConstArray)
5646: end;
5647:
5648: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5649: begin
5650:  Function HexBS( B : Byte) : ShortString
5651:  Function HexWS( W : Word) : ShortString
5652:  Function HexLS( L : LongInt) : ShortString
5653:  Function HexPtrS( P : Pointer) : ShortString
5654:  Function BinaryBS( B : Byte) : ShortString
5655:  Function BinaryWS( W : Word) : ShortString
5656:  Function BinaryLS( L : LongInt) : ShortString
5657:  Function OctalBS( B : Byte) : ShortString
5658:  Function OctalWS( W : Word) : ShortString
5659:  Function OctalLS( L : LongInt) : ShortString
5660:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
```

```
5661:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5662:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5663:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5664:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5665:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5666:  Function Long2StrS( L : LongInt) : ShortString
5667:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5668:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5669:  Function ValPrepS( const S : ShortString) : ShortString
5670:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5671:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5672:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5673:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5674:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5675:  Function TrimLeadS( const S : ShortString) : ShortString
5676:  Function TrimTrailS( const S : ShortString) : ShortString
5677:  Function TrimS( const S : ShortString) : ShortString
5678:  Function TrimSpacesS( const S : ShortString) : ShortString
5679:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5680:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5681:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5682:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5683:  Function ScrambleS( const S, Key : ShortString) : ShortString
5684:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5685:  Function FilterS( const S, Filters : ShortString) : ShortString
5686:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5687:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5688:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5689:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5690:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5691:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5692:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5693:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5694:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
5695:  Function CompStringS( const S1, S2 : ShortString) : Integer
5696:  Function CompUCStringS( const S1, S2 : ShortString) : Integer
5697:  Function SoundexS( const S : ShortString) : ShortString
5698:  Function MakeLetterSetS( const S : ShortString) : Longint
5699:  Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5700:  Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5701:  Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
      Pos:Cardinal):Bool;
5702:  Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5703:  Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5704:  Function JustFilenameS( const PathName : ShortString) : ShortString
5705:  Function JustNameS( const PathName : ShortString) : ShortString
5706:  Function JustExtensionS( const Name : ShortString) : ShortString
5707:  Function JustPathnameS( const PathName : ShortString) : ShortString
5708:  Function AddBackSlashS( const DirName : ShortString) : ShortString
5709:  Function CleanPathNameS( const PathName : ShortString) : ShortString
5710:  Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5711:  Function CommaizeS( L : LongInt) : ShortString
5712:  Function CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5713:  Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
      DecPt:Char):ShortString;
5714:  Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5715:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5716:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5717:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5718:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5719:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5720:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5721:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5722:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5723:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5724:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5725:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5726:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5727:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5728:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
      SubString:ShortString):Bool;
5729:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
      SubStr:ShortString):Bool;
5730:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5731:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5732:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5733:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5734:  Function ExtractTokensS(const S,
      Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5735:  Function IsChAlphaS( C : Char) : Boolean
5736:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5737:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5738:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5739:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
```

```
5740:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5741:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5742:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5743:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5744:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5745:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5746:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
      Cardinal):ShortString;
5747:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
      Replacements:Cardinal):ShortString;
5748:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5749:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
      Replacements:Cardinal):ShortString
5750:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
      Replacements:Cardinal):ShortString
5751:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5752:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5753:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5754:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5755: end;
5756:
5757:
5758: ********unit uPSI_StUtils; from Systools4*****************************************************
5759: Function SignL( L : LongInt) : Integer
5760: Function SignF( F : Extended) : Integer
5761: Function MinWord( A, B : Word) : Word
5762: Function MidWord( W1, W2, W3 : Word) : Word
5763: Function MaxWord( A, B : Word) : Word
5764: Function MinLong( A, B : LongInt) : LongInt
5765: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5766: Function MaxLong( A, B : LongInt) : LongInt
5767: Function MinFloat( F1, F2 : Extended) : Extended
5768: Function MidFloat( F1, F2, F3 : Extended) : Extended
5769: Function MaxFloat( F1, F2 : Extended) : Extended
5770: Function MakeInteger16( H, L : Byte) : SmallInt
5771: Function MakeWordS( H, L : Byte) : Word
5772: Function SwapNibble( B : Byte) : Byte
5773: Function SwapWord( L : LongInt) : LongInt
5774: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5775: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5776: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5777: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5778: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5779: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5780: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5781: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5782: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5783: Procedure ExchangeBytes( var I, J : Byte)
5784: Procedure ExchangeWords( var I, J : Word)
5785: Procedure ExchangeLongInts( var I, J : LongInt)
5786: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5787: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5788: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5789: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5790: //*****************uPSI_StFIN;*************************************************
5791: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5792: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
      Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5793: Function BondDuration( Settlement,Maturity:TStDate;Rate,
      Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5794: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
      Extended
5795: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5796: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5797: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5798: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5799: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5800: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5801: Function DollarToDecimalText( DecDollar : Extended) : string
5802: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5803: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5804: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5805: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5806: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5807: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5808: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5809: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5810: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5811: Function IsCardValid( const S : string) : Boolean
5812: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5813: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5814: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5815: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5816: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
```

```
5817: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5818: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5819: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5820: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5821: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5822: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5823: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5824: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5825: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5826: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5827: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5828: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5829: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5830: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5831: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5832:
5833: //*****************************************unit uPSI_StAstroP;
5834: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5835: //*****unit unit uPSI_StStat; Statistic Package of SysTools***************************************
5836: Function AveDev( const Data : array of Double) : Double
5837: Function AveDev16( const Data, NData : Integer) : Double
5838: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5839: Function Correlation( const Data1, Data2 : array of Double) : Double
5840: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5841: Function Covariance( const Data1, Data2 : array of Double) : Double
5842: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5843: Function DevSq( const Data : array of Double) : Double
5844: Function DevSq16( const Data, NData : Integer) : Double
5845: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5846:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5847: Function GeometricMeanS( const Data : array of Double) : Double
5848: Function GeometricMean16( const Data, NData : Integer) : Double
5849: Function HarmonicMeanS( const Data : array of Double) : Double
5850: Function HarmonicMean16( const Data, NData : Integer) : Double
5851: Function Largest( const Data : array of Double; K : Integer) : Double
5852: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5853: Function MedianS( const Data : array of Double) : Double
5854: Function Median16( const Data, NData : Integer) : Double
5855: Function Mode( const Data : array of Double) : Double
5856: Function Mode16( const Data, NData : Integer) : Double
5857: Function Percentile( const Data : array of Double; K : Double) : Double
5858: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5859: Function PercentRank( const Data : array of Double; X : Double) : Double
5860: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5861: Function Permutations( Number, NumberChosen : Integer) : Extended
5862: Function Combinations( Number, NumberChosen : Integer) : Extended
5863: Function FactorialS( N : Integer) : Extended
5864: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5865: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5866: Function Smallest( const Data : array of Double; K : Integer) : Double
5867: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5868: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5869: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5870:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5871:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5872: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5873: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5874: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5875: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5876: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5877: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5878: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5879: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5880: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5881: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5882: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5883: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5884: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5885: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5886: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5887: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5888: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5889: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5890: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5891: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5892: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5893: Function NormSDist( Z : Single) : Single
5894: Function NormSInv( Probability : Single) : Single
5895: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5896: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5897: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5898: Function Erfc( X : Single) : Single
5899: Function GammaLn( X : Single) : Single
```

```
5900: Function LargestSort( const Data : array of Double; K : Integer) : Double
5901: Function SmallestSort( const Data : array of double; K : Integer) : Double
5902:
5903: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5904:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5905:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5906:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5907:  Function DefaultMergeName( MergeNum : Integer) : string
5908:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5909:
5910: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5911: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5912:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5913:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5914:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5915:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5916:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5917:  Function LunarPhase( UT : TStDateTimeRec) : Double
5918:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5919:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5920:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5921:  Function FullMoon( D : TStDate) : TStLunarRecord
5922:  Function LastQuarter( D : TStDate) : TStLunarRecord
5923:  Function NewMoon( D : TStDate) : TStLunarRecord
5924:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5925:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5926:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5927:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5928:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5929:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5930:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5931:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5932:  Function SiderealTime( UT : TStDateTimeRec) : Double
5933:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5934:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5935:  Function SEaster( Y, Epoch : Integer) : TStDate
5936:  Function DateTimeToAJD( D : TDateTime) : Double
5937:  Function HoursMin( RA : Double) : ShortString
5938:  Function DegsMin( DC : Double) : ShortString
5939:  Function AJDToDateTime( D : Double) : TDateTime
5940:
5941: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5942:  Function CurrentDate : TStDate
5943:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5944:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5945:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5946:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5947:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5948:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5949:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5950:  Function WeekOfYear( Julian : TStDate) : Byte
5951:  Function AstJulianDate( Julian : TStDate) : Double
5952:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5953:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5954:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5955:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5956:  Function StIsLeapYear( Year : Integer) : Boolean
5957:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5958:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5959:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5960:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5961:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5962:  Function CurrentTime : TStTime
5963:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5964:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5965:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5966:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5967:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5968:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5969:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5970:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5971:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5972:  Function StDateToDateTime( D : TStDate) : TDateTime
5973:  Function StTimeToDateTime( T : TStTime) : TDateTime
5974:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5975:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5976:
5977: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5978: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5979:  Function MonthToString( const Month : Integer) : string
5980:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5981:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5982:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5983:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5984:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5985:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5986:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5987:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5988:  Function TimeStringToStTime( const Picture, S : string) : TStTime
```

```
5989:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5990:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5991:  Function DateStringIsBlank( const Picture, S : string) : Boolean
5992:  Function InternationalDate( ForceCentury : Boolean) : string
5993:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5994:  Function InternationalTime( ShowSeconds : Boolean) : string
5995:  Procedure ResetInternationalInfo
5996:
5997: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5998:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5999:  Function AnsiUpperCaseShort32( const S : string) : string
6000:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
6001:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
6002:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
6003:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
6004:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
6005:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
6006:  Function Upcase( C : AnsiChar) : AnsiChar
6007:  Function LoCase( C : AnsiChar) : AnsiChar
6008:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
6009:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
6010:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
6011:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6012:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
6013:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
6014:  Procedure RaiseContainerError( Code : longint)
6015:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
6016:  Function ProductOverflow( A, B : LongInt) : Boolean
6017:  Function StNewStr( S : string) : PShortString
6018:  Procedure StDisposeStr( PS : PShortString)
6019:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
6020:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
6021:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
6022:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
6023:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
6024:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
6025:
6026: procedure SIRegister_usvd(CL: TPSPascalCompiler);
6027: begin
6028:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
6029:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
6030:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
6031:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
6032:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
6033: end;
6034:
6035: //**********unit unit ; StMath Package of SysTools*****************************************
6036: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
6037: Function PowerS( Base, Exponent : Extended) : Extended
6038: Function StInvCos( X : Double) : Double
6039: Function StInvSin( Y : Double) : Double
6040: Function StInvTan2( X, Y : Double) : Double
6041: Function StTan( A : Double) : Double
6042: Procedure DumpException;  //unit StExpEng;
6043: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6044:
6045: //**********unit unit ; StCRC Package of SysTools*****************************************
6046: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6047: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6048: Function Adler32OfFile( FileName : AnsiString) : LongInt
6049: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6050: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6051: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6052: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6053: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6054: Function Crc32OfFile( FileName : AnsiString) : LongInt
6055: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6056: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6057: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6058: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6059: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6060: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6061:
6062: //**********unit unit ; StBCD Package of SysTools*****************************************
6063: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6064: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6065: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6066: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6067: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6068: Function NegBcd( const B : TbcdS) : TbcdS
6069: Function AbsBcd( const B : TbcdS) : TbcdS
6070: Function FracBcd( const B : TbcdS) : TbcdS
6071: Function IntBcd( const B : TbcdS) : TbcdS
6072: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6073: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6074: Function ValBcd( const S : string) : TbcdS
6075: Function LongBcd( L : LongInt) : TbcdS
6076: Function ExtBcd( E : Extended) : TbcdS
6077: Function ExpBcd( const B : TbcdS) : TbcdS
```

```
6078: Function LnBcd( const B : TbcdS) : TbcdS
6079: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6080: Function PowBcd( const B, E : TbcdS) : TbcdS
6081: Function SqrtBcd( const B : TbcdS) : TbcdS
6082: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6083: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6084: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6085: Function IsIntBcd( const B : TbcdS) : Boolean
6086: Function TruncBcd( const B : TbcdS) : LongInt
6087: Function BcdExt( const B : TbcdS) : Extended
6088: Function RoundBcd( const B : TbcdS) : LongInt
6089: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6090: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6091: Function FormatBcd( const Format : string; const B : TbcdS) : string
6092: Function StrGeneralBcd( const B : TbcdS) : string
6093: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6094: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6095:
6096: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6097: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6098: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6099: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6100: Function StDeEscape( const EscStr : AnsiString) : Char
6101: Function StDoEscape( Delim : Char) : AnsiString
6102: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6103: Function AnsiHashText( const S : string; Size : Integer) : Integer
6104: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6105: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6106: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6107:
6108: //**********unit unit ; StNetCon Package of SysTools*****************************************
6109:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6110:     Constructor Create( AOwner : TComponent)
6111:     Function Connect : DWord
6112:     Function Disconnect : DWord
6113:     RegisterProperty('Password', 'String', iptrw);
6114:     Property('UserName', 'String', iptrw);
6115:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6116:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6117:     Property('LocalDevice', 'String', iptrw);
6118:     Property('ServerName', 'String', iptrw);
6119:     Property('ShareName', 'String', iptrw);
6120:     Property('OnConnect', 'TNotifyEvent', iptrw);
6121:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6122:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6123:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6124:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6125:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6126:   end;
6127: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6128: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6129: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6130: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6131: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6132: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6133: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6134: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6135: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6136: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6137: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6138: Function SuspendThread( hThread : THandle) : DWORD
6139: Function ResumeThread( hThread : THandle) : DWORD
6140: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6141: Function GetCurrentThread : THandle
6142: Procedure ExitThread( dwExitCode : DWORD)
6143: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6144: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6145: Procedure EndThread(ExitCode: Integer);
6146: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6147: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6148: Procedure FreeProcInstance( Proc : FARPROC)
6149: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6150: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6151: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6152: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6153: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6154: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6155: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6156: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6157: Function CurrentParallelJobInfo : TParallelJobInfo
6158: Function ObtainParallelJobInfo : TParallelJobInfo
6159: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6160: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL');
6161: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL');
6162: Function
      DeviceIoControl(hDevice:THandle;dwIoControlCode:DWORD;lpInBuffer:TObject;nInBufferSize:DWORD;lpOutBuffer:
      TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped:TOverlapped):BOOL');
6163: Function SetFileTime(hFile:THandle;lpCreationTime,lpLastAccessTime,lpLastWriteTime:TFileTime): BOOL');
```

```
6164: Function DuplicateHandle(hSourceProcessHandle,hSourceHandle,hTargetProcessHandle:THandle;
      lpTargetHandle:THandle; dwDesiredAccess : DWORD; bInheritHandle:BOOL; dwOptions : DWORD) : BOOL');
6165: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL');
6166: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL');
6167:
6168: ************************************************unit uPSI_JclMime;
6169:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6170:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6171:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6172:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6173:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6174:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6175:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6176:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6177:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6178:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6179:
6180: ************************************************unit uPSI_JclPrint;
6181:  Procedure DirectPrint( const Printer, Data : string)
6182:  Procedure SetPrinterPixelsPerInch
6183:  Function GetPrinterResolution : TPoint
6184:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6185:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6186:
6187:
6188: //*************************************unit uPSI_ShLwApi;************************************
6189:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6190:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6191:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6192:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6193:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6194:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6195:  Function StrDup( lpSrch : PChar) : PChar
6196:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6197:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6198:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6199:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6200: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6201:  Function StrPBrk( psz, pszSet : PChar) : PChar
6202:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6203:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6204:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6205:  Function StrSpn( psz, pszSet : PChar) : Integer
6206:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6207:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6208:  Function StrToInt( lpSrch : PChar) : Integer
6209:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6210:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6211:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6212:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6213:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6214:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6215:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6216:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6217:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6218:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6219:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6220:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6221:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6222:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6223:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6224:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6225:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6226:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6227:  STIF_DEFAULT','LongWord( $00000000);
6228:  STIF_SUPPORT_HEX','LongWord( $00000001);
6229:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6230:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6231:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6232:  Function PathAddBackslash( pszPath : PChar) : PChar
6233:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6234:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6235:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6236:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6237:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6238:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6239:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6240:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6241:  Function PathFileExists( pszPath : PChar) : BOOL
6242:  Function PathFindExtension( pszPath : PChar) : PChar
6243:  Function PathFindFileName( pszPath : PChar) : PChar
6244:  Function PathFindNextComponent( pszPath : PChar) : PChar
6245:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6246:  Function PathGetArgs( pszPath : PChar) : PChar
6247:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6248:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6249:  Function PathGetCharType( ch : Char) : UINT
```

```
6250:  GCT_INVALID','LongWord( $0000);
6251:  GCT_LFNCHAR','LongWord( $0001);
6252:  GCT_SHORTCHAR','LongWord( $0002);
6253:  GCT_WILD','LongWord( $0004);
6254:  GCT_SEPARATOR','LongWord( $0008);
6255:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6256:  Function PathIsDirectory( pszPath : PChar) : BOOL
6257:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6258:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6259:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6260:  Function PathIsRelative( pszPath : PChar) : BOOL
6261:  Function PathIsRoot( pszPath : PChar) : BOOL
6262:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6263:  Function PathIsUNC( pszPath : PChar) : BOOL
6264:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6265:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6266:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6267:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6268:  Function PathIsURL( pszPath : PChar) : BOOL
6269:  Function PathMakePretty( pszPath : PChar) : BOOL
6270:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6271:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6272:  Procedure PathQuoteSpaces( lpsz : PChar)
6273:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6274:  Procedure PathRemoveArgs( pszPath : PChar)
6275:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6276:  Procedure PathRemoveBlanks( pszPath : PChar)
6277:  Procedure PathRemoveExtension( pszPath : PChar)
6278:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6279:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6280:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6281:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6282:  Function PathSkipRoot( pszPath : PChar) : PChar
6283:  Procedure PathStripPath( pszPath : PChar)
6284:  Function PathStripToRoot( pszPath : PChar) : BOOL
6285:  Procedure PathUnquoteSpaces( lpsz : PChar)
6286:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6287:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6288:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6289:  Procedure PathUndecorate( pszPath : PChar)
6290:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6291:  URL_SCHEME_INVALID','LongInt'( - 1);
6292:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6293:  URL_SCHEME_FTP','LongInt'( 1);
6294:  URL_SCHEME_HTTP','LongInt'( 2);
6295:  URL_SCHEME_GOPHER','LongInt'( 3);
6296:  URL_SCHEME_MAILTO','LongInt'( 4);
6297:  URL_SCHEME_NEWS','LongInt'( 5);
6298:  URL_SCHEME_NNTP','LongInt'( 6);
6299:  URL_SCHEME_TELNET','LongInt'( 7);
6300:  URL_SCHEME_WAIS','LongInt'( 8);
6301:  URL_SCHEME_FILE','LongInt'( 9);
6302:  URL_SCHEME_MK','LongInt'( 10);
6303:  URL_SCHEME_HTTPS','LongInt'( 11);
6304:  URL_SCHEME_SHELL','LongInt'( 12);
6305:  URL_SCHEME_SNEWS','LongInt'( 13);
6306:  URL_SCHEME_LOCAL','LongInt'( 14);
6307:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6308:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6309:  URL_SCHEME_ABOUT','LongInt'( 17);
6310:  URL_SCHEME_RES','LongInt'( 18);
6311:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6312:  URL_SCHEME', 'Integer
6313:  URL_PART_NONE','LongInt'( 0);
6314:  URL_PART_SCHEME','LongInt'( 1);
6315:  URL_PART_HOSTNAME','LongInt'( 2);
6316:  URL_PART_USERNAME','LongInt'( 3);
6317:  URL_PART_PASSWORD','LongInt'( 4);
6318:  URL_PART_PORT','LongInt'( 5);
6319:  URL_PART_QUERY','LongInt'( 6);
6320:  URL_PART', 'DWORD
6321:  URLIS_URL','LongInt'( 0);
6322:  URLIS_OPAQUE','LongInt'( 1);
6323:  URLIS_NOHISTORY','LongInt'( 2);
6324:  URLIS_FILEURL','LongInt'( 3);
6325:  URLIS_APPLIABLE','LongInt'( 4);
6326:  URLIS_DIRECTORY','LongInt'( 5);
6327:  URLIS_HASQUERY','LongInt'( 6);
6328:  TUrlIs', 'DWORD
6329:  URL_UNESCAPE','LongWord( $10000000);
6330:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6331:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6332:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6333:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6334:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6335:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6336:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6337:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6338:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
```

```
6339:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6340:  URL_INTERNAL_PATH','LongWord( $00800000);
6341:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6342:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6343:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6344:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6345:  URL_APPLY_DEFAULT','LongWord( $00000001);
6346:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6347:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6348:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6349:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6350:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6351:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6352:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6353:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6354:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6355:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6356:  Function UrlGetLocation( psz1 : PChar) : PChar
6357:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6358:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6359:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6360:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6361:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6362:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6363:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6364:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6365:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6366:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6367:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6368:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6369:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6370:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6371:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
      pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6372:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6373:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6374:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6375:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6376:  SHREGDEL_DEFAULT','LongWord( $00000000);
6377:  SHREGDEL_HKCU','LongWord( $00000001);
6378:  SHREGDEL_HKLM','LongWord( $00000010);
6379:  SHREGDEL_BOTH','LongWord( $00000011);
6380:  SHREGENUM_DEFAULT','LongWord( $00000000);
6381:  SHREGENUM_HKCU','LongWord( $00000001);
6382:  SHREGENUM_HKLM','LongWord( $00000010);
6383:  SHREGENUM_BOTH','LongWord( $00000011);
6384:  SHREGSET_HKCU','LongWord( $00000001);
6385:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6386:  SHREGSET_HKLM','LongWord( $00000004);
6387:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6388:  TSHRegDelFlags', 'DWORD
6389:  TSHRegEnumFlags', 'DWORD
6390:  HUSKEY', 'THandle
6391:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6392:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6393:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6394:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6395:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6396:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6397:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6398:  ASSOCF_VERIFY','LongWord( $00000040);
6399:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6400:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6401:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6402:  ASSOCF', 'DWORD
6403:  ASSOCSTR_COMMAND','LongInt'( 1);
6404:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6405:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6406:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6407:  ASSOCSTR_NOOPEN','LongInt'( 5);
6408:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6409:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
6410:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6411:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6412:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6413:  ASSOCSTR_INFOTIP','LongInt'( 11);
6414:  ASSOCSTR_MAX','LongInt'( 12);
6415:  ASSOCSTR', 'DWORD
6416:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6417:  ASSOCKEY_APP','LongInt'( 2);
6418:  ASSOCKEY_CLASS','LongInt'( 3);
6419:  ASSOCKEY_BASECLASS','LongInt'( 4);
6420:  ASSOCKEY_MAX','LongInt'( 5);
6421:  ASSOCKEY', 'DWORD
6422:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6423:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6424:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6425:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6426:  ASSOCDATA_MAX','LongInt'( 5);
```

```
6427:  ASSOCDATA', 'DWORD
6428:  ASSOCENUM_NONE','LongInt'( 0);
6429:  ASSOCENUM', 'DWORD
6430:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6431:  SHACF_DEFAULT $00000000);
6432:  SHACF_FILESYSTEM','LongWord( $00000001);
6433:  SHACF_URLHISTORY','LongWord( $00000002);
6434:  SHACF_URLMRU','LongWord( $00000004);
6435:  SHACF_USETAB','LongWord( $00000008);
6436:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6437:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6438:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6439:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6440:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6441:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6442:  Procedure SHSetThreadRef( punk : IUnknown)
6443:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6444:  CTF_INSIST','LongWord( $00000001);
6445:  CTF_THREAD_REF','LongWord( $00000002);
6446:  CTF_PROCESS_REF','LongWord( $00000004);
6447:  CTF_COINIT','LongWord( $00000008);
6448:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6449:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6450:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6451:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6452:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6453:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6454:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6455:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6456:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6457:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6458:  Function SetRectEmpty( var lprc : TRect) : BOOL
6459:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6460:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6461:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6462:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6463:
6464:  Function InitializeFlatSB( hWnd : HWND) : Bool
6465:  Procedure UninitializeFlatSB( hWnd : HWND)
6466:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6467:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6468:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6469:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6470:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6471:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6472:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6473:
6474:
6475: //  ***************************************** 204 unit uPSI_ShellAPI;
6476: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6477: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6478: Procedure DragFinish( Drop : HDROP)
6479: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6480: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6481: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6482: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6483: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6484: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6485: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6486: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6487: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6488: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6489: Procedure SHFreeNameMappings( hNameMappings : THandle)
6490:
6491:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6492:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6493:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6494:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6495:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6496:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6497:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6498:  Function SimpleXMLEncode( const S : string) : string
6499:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6500:  Function XMLEncode( const S : string) : string
6501:  Function XMLDecode( const S : string) : string
6502:  Function EntityEncode( const S : string) : string
6503:  Function EntityDecode( const S : string) : string
6504:
6505: procedure RIRegister_CPort_Routines(S: TPSExec);
6506:  Procedure EnumComPorts( Ports : TStrings)
6507:  Procedure ListComPorts( Ports : TStrings)
6508:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6509:  Function GetComPorts: TStringlist;
6510:  Function StrToBaudRate( Str : string) : TBaudRate
6511:  Function StrToStopBits( Str : string) : TStopBits
6512:  Function StrToDataBits( Str : string) : TDataBits
6513:  Function StrToParity( Str : string) : TParityBits
6514:  Function StrToFlowControl( Str : string) : TFlowControl
6515:  Function BaudRateToStr( BaudRate : TBaudRate) : string
```

```
6516:  Function StopBitsToStr( StopBits : TStopBits) : string
6517:  Function DataBitsToStr( DataBits : TDataBits) : string
6518:  Function ParityToStr( Parity : TParityBits) : string
6519:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6520:  Function ComErrorsToStr( Errors : TComErrors) : String
6521:
6522:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6523:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6524:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6525:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6526:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6527:  Function GetMessagePos : DWORD
6528:  Function GetMessageTime : Longint
6529:  Function GetMessageExtraInfo : Longint
6530:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6531:  Procedure JAddToRecentDocs( const Filename : string)
6532:  Procedure ClearRecentDocs
6533:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6534:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6535:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6536:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6537:  Function RecycleFile( FileToRecycle : string) : Boolean
6538:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6539:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6540:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6541:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6542:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6543:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6544:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6545:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6546:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6547:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6548:
6549: ***************************************** unit uPSI_JclPeImage;
6550:
6551:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6552: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6553:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6554:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
6555:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6556:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6557:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6558:  Function PeDoesExportFunction(const FileName:TFileName;const
      FuncName:string;Options:TJclSmartCompOptions):Bool;
6559:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6560:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
      FunctionName:string;Options:TJclSmartCompOptions):Bool
6561:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean
6562:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
6563:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean
6564:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
      LibraryName:string; IncludeLibNames : Boolean): Boolean
6565:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6566:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6567:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6568:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
      NamesList:TStrings):Bool
6569:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6570:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
      Descript:Bool):Bool;
6571:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6572:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6573:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6574:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6575:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6576:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6577:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
      PImageSectionHeader
6578:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6579:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6580:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
      ___Pointer;
6581:   SIRegister_TJclPeSectionStream(CL);
6582:   SIRegister_TJclPeMapImgHookItem(CL);
6583:   SIRegister_TJclPeMapImgHooks(CL);
```

```
6584:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
       NtHeaders:TImageNtHeaders):Boolean
6585:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6586:   Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6587:   TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6588:   TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6589:   TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6590:   TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6591:   TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6592:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
       TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6593:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6594:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6595:  Function PeBorUnmangleName3( const Name : string) : string;
6596:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6597:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6598:
6599:
6600: //***************** SysTools uPSI_StSystem; ************************************
6601:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6602:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6603:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6604:  //Procedure EnumerateDirectories(const
       StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6605:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
       IncludeItem:TIncludeItemFunc);
6606:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6607:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6608:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6609:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6610:  Function FlushOsBuffers( Handle : Integer) : Boolean
6611:  Function GetCurrentUser : AnsiString
6612:  Function GetDiskClass( Drive : Char) : DiskClass
6613:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
       SectorsPerCluster:Cardinal):Bool;
6614:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
       DiskSize:Double):Bool;
6615:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
       DiskSize:Comp):Boolean;
6616:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6617:  Function getDiskSpace2(const path: String; index: integer): int64;
6618:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6619:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6620:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6621:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6622:  Function GetLongPath( const APath : AnsiString) : AnsiString
6623:  Function GetMachineName : AnsiString
6624:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6625:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6626:  Function GetShortPath( const APath : AnsiString) : AnsiString
6627:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6628:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6629:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6630:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6631:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6632:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6633:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6634:  Function IsDriveReady( Drive : Char) : Boolean
6635:  Function IsFile( const FileName : AnsiString) : Boolean
6636:  Function IsFileArchive( const S : AnsiString) : Integer
6637:  Function IsFileHidden( const S : AnsiString) : Integer
6638:  Function IsFileReadOnly( const S : AnsiString) : Integer
6639:  Function IsFileSystem( const S : AnsiString) : Integer
6640:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6641:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6642:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6643:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6644:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6645:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6646:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6647:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6648:  Function ValidDrive( Drive : Char) : Boolean
6649:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6650:
6651: //****************************unit uPSI_JclLANMan;************************************
6652: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6653:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6654:  Function DeleteAccount( const Servername, Username : string) : Boolean
6655:  Function DeleteLocalAccount( Username : string) : Boolean
6656:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6657:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6658:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6659:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6660:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6661:  Function LocalGroupExists( const Group : string) : Boolean
6662:  Function GlobalGroupExists( const Server, Group : string) : Boolean
```

```
6663:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6664:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6665:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6666:  Function IsLocalAccount( const AccountName : string) : Boolean
6667:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6668:  Function GetRandomString( NumChar : cardinal) : string
6669:
6670: //****************************unit uPSI_cUtils;****************************************
6671: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6672: Function cIsWinNT : boolean
6673:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6674:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6675:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6676:  Function cGetShortName( FileName : string) : string
6677:  Procedure cShowError( Msg : String)
6678:  Function cCommaStrToStr( s : string; formatstr : string) : string
6679:  Function cIncludeQuoteIfSpaces( s : string) : string
6680:  Function cIncludeQuoteIfNeeded( s : string) : string
6681:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6682:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6683:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6684:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6685:  Function cCodeInstoStr( s : string) : string
6686:  Function cStrtoCodeIns( s : string) : string
6687:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6688:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6689:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6690:  Function cPointtoStr( const pt : TPoint) : string
6691:  Function cListtoStr( const List : TStrings) : string
6692:  Function ListtoStr( const List : TStrings) : string
6693:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6694:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6695:  Function cGetFileTyp( const FileName : string) : TUnitType
6696:  Function cGetExTyp( const FileName : string) : TExUnitType
6697:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6698:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6699:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6700:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6701:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6702:  Function cGenMakePath( FileName : String) : String;
6703:  Function cGenMakePath2( FileName : String) : String
6704:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6705:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6706:  Function cCalcMod( Count : Integer) : Integer
6707:  Function cGetVersionString( FileName : string) : string
6708:  Function cCheckChangeDir( var Dir : string) : boolean
6709:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6710:  Function cIsNumeric( s : string) : boolean
6711:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6712:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6713:  Function GetFileTyp( const FileName : string) : TUnitType
6714:  Function Atoi(const aStr: string): integer
6715:  Function Itoa(const aint: integer): string
6716:  Function Atof(const aStr: string): double');
6717:  Function Atol(const aStr: string): longint');
6718:
6719:
6720: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6721: begin
6722:   FindClass('TOBJECT'),'EHTTP
6723:   FindClass('TOBJECT'),'EHTTPParser
6724:   //AnsiCharSet', 'set of AnsiChar
6725:   AnsiStringArray', 'array of AnsiString
6726:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6727:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6728:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6729:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6730:    +'CustomMinVersion : Integer; end
6731:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6732:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6733:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6734:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6735:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6736:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6737:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6738:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6739:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6740:    +'nection, hntOrigin, hntKeepAlive )
6741:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6742:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6743:    +' AnsiString; end
6744:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6745:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6746:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6747:   //PHTTPContentLength', '^THTTPContentLength // will not work
6748:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6749:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
```

```
6750:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6751:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6752:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6753:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6754:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6755:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6756:    +' CustomStr : AnsiString; end
6757:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6758:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6759:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6760:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6761:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6762:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6763:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6764:    +'m; Custom : AnsiString; end
6765:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6766:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6767:    +' Custom : AnsiString; end
6768:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6769:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6770:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6771:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6772:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6773:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6774:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6775:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6776:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6777:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6778:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6779:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6780:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6781:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6782:    +'FieldEnum; List : array of THTTPContentEncoding; end
6783:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6784:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6785:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6786:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6787:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6788:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6789:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6790:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6791:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6792:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6793:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6794:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6795:    +'CustomFieldArray; Custom : AnsiString; end
6796:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6797:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6798:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6799:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6800:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6801:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6802:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6803:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6804:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6805:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6806:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6807:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6808:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6809:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6810:   THTTPFixedHeaders','array[0..42] of AnsiString
6811:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6812:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6813:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6814:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6815:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6816:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6817:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6818:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6819:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6820:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6821:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6822:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6823:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6824:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6825:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6826:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6827:    +' THTTPDateField; Age : THTTPAgeField; end
6828:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6829:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6830:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6831:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6832:  Procedure InitHTTPRequest( var A : THTTPRequest)
6833:  Procedure InitHTTPResponse( var A : THTTPResponse)
6834:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6835:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6836:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6837:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6838:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
```

```
6839:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6840:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6841:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6842:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6843:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6844:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6845:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6846:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6847:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6848:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6849:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6850:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6851:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6852:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6853:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6854:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6855:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6856:   THTTPStringOption', '( hsoNone )
6857:   THTTPStringOptions', 'set of THTTPStringOption
6858:   FindClass('TOBJECT'),'TAnsiStringBuilder
6859:
6860:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6861:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6862:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6863:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6864:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6865:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6866:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6867:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6868:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6869:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6870:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6871:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6872:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6873:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6874:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6875:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
6876:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6877:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6878:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions)
6879:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6880:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6881:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6882:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6883:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6884:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6885:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6886:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6887:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6888:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
      THTTPStrOptions);
6889:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6890:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
      P:THTTPStringOptions);
6891:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6892:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6893:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6894:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6895:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6896:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6897:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
      Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
```

```
6898:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6899:   SIRegister_THTTPParser(CL);
6900:   FindClass('TOBJECT'),'THTTPContentDecoder
6901:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6902:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6903:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6904:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6905:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6906:   SIRegister_THTTPContentDecoder(CL);
6907:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6908:   FindClass('TOBJECT'),'THTTPContentReader
6909:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6910:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
      LogLevel:Int;
6911:   SIRegister_THTTPContentReader(CL);
6912:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6913:   FindClass('TOBJECT'),'THTTPContentWriter
6914:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6915:   SIRegister_THTTPContentWriter(CL);
6916:  Procedure SelfTestcHTTPUtils
6917: end;
6918:
6919: (*----------------------------------------------------------------------------*)
6920: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6921: begin
6922:  'TLSLibraryVersion','String '1.00
6923:  'TLSError_None','LongInt'( 0);
6924:  'TLSError_InvalidBuffer','LongInt'( 1);
6925:  'TLSError_InvalidParameter','LongInt'( 2);
6926:  'TLSError_InvalidCertificate','LongInt'( 3);
6927:  'TLSError_InvalidState','LongInt'( 4);
6928:  'TLSError_DecodeError','LongInt'( 5);
6929:  'TLSError_BadProtocol','LongInt'( 6);
6930:  Function TLSErrorMessage( const TLSError : Integer) : String
6931:   SIRegister_ETLSError(CL);
6932:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6933:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6934:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6935:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6936:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6937:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6938:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6939:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6940:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6941:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6942:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6943:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6944:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6945:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6946:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6947:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6948:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6949:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6950:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6951:   PTLSRandom', '^TTLSRandom // will not work
6952:  Procedure InitTLSRandom( var Random : TTLSRandom)
6953:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6954:  'TLSSessionIDMaxLen','LongInt'( 32);
6955:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6956:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6957:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6958:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6959:    +'; Signature : TTLSSignatureAlgorithm; end
6960:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6961:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6962:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6963:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6964:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6965:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6966:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6967:    +'nteger; Supported : Boolean; end
6968:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6969:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
6970:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6971:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6972:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6973:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6974:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6975:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6976:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6977:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6978:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
6979:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
6980:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6981:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
```

```
6982:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
6983:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6984:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6985:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6986:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
6987:    TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6988:     +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6989:     +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6990:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
      IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6991:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
      ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6992:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
6993:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
6994:  Procedure SelfTestcTLSUtils
6995: end;
6996:
6997: (*----------------------------------------------------------------------------*)
6998: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6999: begin
7000:    sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
      integer; disks : integer; mx : integer; my : integer; end
7001:  // pBoard', '^tBoard // will not work
7002:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
7003:  Function rCheckMove( color : byte; cx, cy : integer) : integer
7004:  //Function rDoStep( data : pBoard) : word
7005:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
7006: end;
7007:
7008: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
7009: begin
7010: Function InEditMode( ADataset : TDataset) : Boolean
7011: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
7012: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
7013: Function GetFieldText( AField : TField) : String
7014: end;
7015:
7016: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
7017: begin
7018:    TPrintMode', '( pmPrint, pmPreview, pmPageCount )
7019:    TMyPrintRange', '( prAll, prSelected )
7020:    TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
7021:     +'ded, ssDateTime, ssTime, ssCustom )
7022:    TSortDirection', '( sdAscending, sdDescending )
7023:    TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
7024:    TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
7025:     +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
7026:    TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
7027:    TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
7028:    SIRegister_TSortOptions(CL);
7029:    SIRegister_TPrintOptions(CL);
7030:    TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
7031:    SIRegister_TSortedList(CL);
7032:    TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
7033:    TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
7034:    TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
7035:    TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
7036:     +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
7037:    SIRegister_TFontSetting(CL);
7038:    SIRegister_TFontList(CL);
7039:    AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
7040:     + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
7041:    TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7042:    TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
7043:    TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
7044:    TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7045:    TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7046:    TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7047:    TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
7048:     +'r; var SortStyle : TSortStyle)
7049:    TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
7050:     +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7051:    SIRegister_TSortGrid(CL);
7052:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
7053:  Function NormalCompare( const Str1, Str2 : String) : Integer
7054:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
7055:  Function NumericCompare( const Str1, Str2 : String) : Integer
7056:  Function TimeCompare( const Str1, Str2 : String) : Integer
7057:  //Function Compare( Item1, Item2 : Pointer) : Integer
7058: end;
7059:
7060: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
7061:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
7062:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
7063:  Procedure IBDataBaseError
7064:  Function StatusVector : PISC_STATUS
7065:  Function StatusVectorArray : PStatusVector
```

```
7066:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
7067:  Function StatusVectorAsText : string
7068:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7069:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7070:
7071:
7072: //*****************************unit uPSI_BoldUtils;*****************************************
7073:  Function CharCount( c : char; const s : string) : integer
7074:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7075:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7076:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7077:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7078:  Function BoldTrim( const S : string) : string
7079:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7080:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7081:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7082:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7083:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7084:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7085:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7086:  Function CapitalisedToSpaced( Capitalised : String) : String
7087:  Function SpacedToCapitalised( Spaced : String) : String
7088:  Function BooleanToString( BoolValue : Boolean) : String
7089:  Function StringToBoolean( StrValue : String) : Boolean
7090:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7091:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7092:  Function StringListToVarArray( List : TStringList) : variant
7093:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7094:  Function GetComputerNameStr : string
7095:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7096:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7097:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7098:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7099:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7100:  Procedure EnsureTrailing( var Str : String; ch : char)
7101:  Function BoldDirectoryExists( const Name : string) : Boolean
7102:  Function BoldForceDirectories( Dir : string) : Boolean
7103:  Function BoldRootRegistryKey : string
7104:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7105:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7106:  Function LogicalAnd( A, B : Integer) : Boolean
7107:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7108:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7109:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7110:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7111:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7112:  Function IsFirstInstance : Boolean
7113:  Procedure ActivateFirst( AString : PChar)
7114:  Procedure ActivateFirstCommandLine
7115:  function MakeAckPkt(const BlockNumber: Word): string;
7116:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7117:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7118:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7119:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7120:  function IdStrToWord(const Value: String): Word;
7121:  function IdWordToStr(const Value: Word): WordStr;
7122:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7123:  Function CPUFeatures : TCPUFeatures
7124:
7125: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7126: begin
7127:   AddTypeS('TXRTLBitIndex', 'Integer
7128:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7129:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7130:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7131:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7132:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7133:  Function XRTLSwapHiLo16( X : Word) : Word
7134:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7135:  Function XRTLSwapHiLo64( X : Int64) : Int64
7136:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7137:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7138:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7139:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7140:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7141:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7142:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7143:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7144:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7145:  Function XRTLPopulation( A : Cardinal) : Cardinal
7146: end;
7147:
7148: Function XRTLURLDecode( const ASrc : WideString) : WideString
7149: Function XRTLURLEncode( const ASrc : WideString) : WideString
7150: Function XRTLURINormalize( const AURI : WideString) : WideString
7151: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7152: Function XRTLExtractLongPathName(APath: string): string;
```

```
7153:
7154: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7155: begin
7156:   AddTypeS('Int8', 'ShortInt
7157:   AddTypeS('Int16', 'SmallInt
7158:   AddTypeS('Int32', 'LongInt
7159:   AddTypeS('UInt8', 'Byte
7160:   AddTypeS('UInt16', 'Word
7161:   AddTypeS('UInt32', 'LongWord
7162:   AddTypeS('UInt64', 'Int64
7163:   AddTypeS('Word8', 'UInt8
7164:   AddTypeS('Word16', 'UInt16
7165:   AddTypeS('Word32', 'UInt32
7166:   AddTypeS('Word64', 'UInt64
7167:   AddTypeS('LargeInt', 'Int64
7168:   AddTypeS('NativeInt', 'Integer
7169:   AddTypeS('NativeUInt', 'Cardinal
7170:   Const('BitsPerByte','LongInt'( 8);
7171:   Const('BitsPerWord','LongInt'( 16);
7172:   Const('BitsPerLongWord','LongInt'( 32);
7173:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7174:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7175:  Function MinI( const A, B : Integer) : Integer
7176:  Function MaxI( const A, B : Integer) : Integer
7177:  Function MinC( const A, B : Cardinal) : Cardinal
7178:  Function MaxC( const A, B : Cardinal) : Cardinal
7179:  Function SumClipI( const A, I : Integer) : Integer
7180:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7181:  Function InByteRange( const A : Int64) : Boolean
7182:  Function InWordRange( const A : Int64) : Boolean
7183:  Function InLongWordRange( const A : Int64) : Boolean
7184:  Function InShortIntRange( const A : Int64) : Boolean
7185:  Function InSmallIntRange( const A : Int64) : Boolean
7186:  Function InLongIntRange( const A : Int64) : Boolean
7187:   AddTypeS('Bool8', 'ByteBool
7188:   AddTypeS('Bool16', 'WordBool
7189:   AddTypeS('Bool32', 'LongBool
7190:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7191:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7192:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7193:  Const('MinSingle','Single').setExtended( 1.5E-45);
7194:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7195:  Const('MinDouble','Double').setExtended( 5.0E-324);
7196:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7197:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7198:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7199:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7200:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7201:  Function MinF( const A, B : Float) : Float
7202:  Function MaxF( const A, B : Float) : Float
7203:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7204:  Function InSingleRange( const A : Float) : Boolean
7205:  Function InDoubleRange( const A : Float) : Boolean
7206:  Function InCurrencyRange( const A : Float) : Boolean;
7207:  Function InCurrencyRange1( const A : Int64) : Boolean;
7208:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7209:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7210:  Function FloatIsInfinity( const A : Extended) : Boolean
7211:  Function FloatIsNaN( const A : Extended) : Boolean
7212:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7213:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7214:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7215:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7216:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7217:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7218:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7219:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7220:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7221:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7222:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7223:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7224:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7225:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7226:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7227:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7228:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7229:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7230:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7231:  Function SetBitScanForward( const Value : LongWord) : Integer;
7232:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7233:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7234:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7235:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7236:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7237:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7238:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7239:  Function cReverseBits( const Value : LongWord) : LongWord;
7240:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7241:  Function cSwapEndian( const Value : LongWord) : LongWord
```

```
7242:  Function cTwosComplement( const Value : LongWord) : LongWord
7243:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7244:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7245:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7246:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7247:  Function cBitCount( const Value : LongWord) : LongWord
7248:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7249:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7250:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7251:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7252:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7253:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7254:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7255:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7256:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7257: //  AddTypeS('CharSet', 'set of AnsiChar
7258:   AddTypeS('CharSet', 'set of Char       //!!!
7259:   AddTypeS('AnsiCharSet', 'TCharSet
7260:   AddTypeS('ByteSet', 'set of Byte
7261:   AddTypeS('AnsiChar', 'Char
7262:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7263:  Function AsByteSet( const C : array of Byte) : ByteSet
7264:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7265:  Procedure ClearCharSet( var C : CharSet)
7266:  Procedure FillCharSet( var C : CharSet)
7267:  procedure FillCharSearchRec;  // with 0
7268:  Procedure ComplementCharSet( var C : CharSet)
7269:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7270:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7271:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7272:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7273:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7274:  Function IsSubSet( const A, B : CharSet) : Boolean
7275:  Function IsEqual( const A, B : CharSet) : Boolean
7276:  Function IsEmpty( const C : CharSet) : Boolean
7277:  Function IsComplete( const C : CharSet) : Boolean
7278:  Function cCharCount( const C : CharSet) : Integer
7279:  Procedure ConvertCaseInsensitive( var C : CharSet)
7280:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7281:  Function IntRangeLength( const Low, High : Integer) : Int64
7282:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7283:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7284:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7285:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7286:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7287:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7288:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7289:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7290:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7291:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7292:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7293:   AddTypeS('UnicodeChar', 'WideChar
7294:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7295:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7296:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7297:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7298:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7299:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7300:  Function cSgn( const A : LongInt) : Integer;
7301:  Function cSgn1( const A : Int64) : Integer;
7302:  Function cSgn2( const A : Extended) : Integer;
7303:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7304:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7305:  Function WideCharToInt( const A : WideChar) : Integer
7306:  Function CharToInt( const A : Char) : Integer
7307:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7308:  Function IntToWideChar( const A : Integer) : WideChar
7309:  Function IntToChar( const A : Integer) : Char
7310:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7311:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7312:  Function IsHexChar( const Ch : Char) : Boolean
7313:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7314:  Function HexWideCharToInt( const A : WideChar) : Integer
7315:  Function HexCharToInt( const A : Char) : Integer
7316:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7317:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7318:  Function IntToUpperHexChar( const A : Integer) : Char
7319:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7320:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7321:  Function IntToLowerHexChar( const A : Integer) : Char
7322:  Function IntToStringA( const A : Int64) : AnsiString
7323:  Function IntToStringW( const A : Int64) : WideString
7324:  Function IntToString( const A : Int64) : String
7325:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7326:  Function UIntToStringW( const A : NativeUInt) : WideString
7327:  Function UIntToString( const A : NativeUInt) : String
7328:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7329:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7330:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
```

```
7331:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7332:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7333:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7334:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7335:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7336:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7337:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7338:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7339:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7340:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7341:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7342:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7343:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7344:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7345:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7346:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7347:  Function StringToInt64A( const S : AnsiString) : Int64
7348:  Function StringToInt64W( const S : WideString) : Int64
7349:  Function StringToInt64( const S : String) : Int64
7350:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7351:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7352:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7353:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7354:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7355:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7356:  Function StringToIntA( const S : AnsiString) : Integer
7357:  Function StringToIntW( const S : WideString) : Integer
7358:  Function StringToInt( const S : String) : Integer
7359:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7360:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7361:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7362:  Function StringToLongWordA( const S : AnsiString) : LongWord
7363:  Function StringToLongWordW( const S : WideString) : LongWord
7364:  Function StringToLongWord( const S : String) : LongWord
7365:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7366:  Function HexToUIntW( const S : WideString) : NativeUInt
7367:  Function HexToUInt( const S : String) : NativeUInt
7368:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7369:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7370:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7371:  Function HexToLongWordA( const S : AnsiString) : LongWord
7372:  Function HexToLongWordW( const S : WideString) : LongWord
7373:  Function HexToLongWord( const S : String) : LongWord
7374:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7375:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7376:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7377:  Function OctToLongWordA( const S : AnsiString) : LongWord
7378:  Function OctToLongWordW( const S : WideString) : LongWord
7379:  Function OctToLongWord( const S : String) : LongWord
7380:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7381:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7382:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7383:  Function BinToLongWordA( const S : AnsiString) : LongWord
7384:  Function BinToLongWordW( const S : WideString) : LongWord
7385:  Function BinToLongWord( const S : String) : LongWord
7386:  Function FloatToStringA( const A : Extended) : AnsiString
7387:  Function FloatToStringW( const A : Extended) : WideString
7388:  Function FloatToString( const A : Extended) : String
7389:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7390:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7391:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7392:  Function StringToFloatA( const A : AnsiString) : Extended
7393:  Function StringToFloatW( const A : WideString) : Extended
7394:  Function StringToFloat( const A : String) : Extended
7395:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7396:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7397:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7398:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
     PadChar: AnsiChar) : AnsiString
7399:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7400:  unit uPSI_cFundamentUtils;
7401:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7402:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7403:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7404:  Const('CCHARSET','Stringb64_XXEncode);
7405:  Const('CHEXSET','String'0123456789ABCDEF
7406:  Const('HEXDIGITS','String'0123456789ABCDEF
7407:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7408:  Const('DIGISET','String'0123456789
7409:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7410:  Const('DIGISET2','TCharset').SetSet('0123456789'
7411:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7412:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7413:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7414:  Const('NUMBERS','String'0123456789');
7415:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7416:  Function CharSetToStr( const C : CharSet) : AnsiString
7417:  Function StrToCharSet( const S : AnsiString) : CharSet
7418:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
```

```
7419:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7420:  Function UUDecode( const S : AnsiString) : AnsiString
7421:  Function XXDecode( const S : AnsiString) : AnsiString
7422:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7423:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7424:  Function InterfaceToStrW( const I : IInterface) : WideString
7425:  Function InterfaceToStr( const I : IInterface) : String
7426:  Function ObjectClassName( const O : TObject) : String
7427:  Function ClassClassName( const C : TClass) : String
7428:  Function ObjectToStr( const O : TObject) : String
7429:  Function ObjectToString( const O : TObject) : String
7430:  Function CharSetToStr( const C : CharSet) : AnsiString
7431:  Function StrToCharSet( const S : AnsiString) : CharSet
7432:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7433:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7434:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7435:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7436:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7437:  Const('Bytes1KB','LongInt'( 1024);
7438:   SIRegister_IInterface(CL);
7439:  Procedure SelfTestCFundamentUtils
7440:
7441: Function CreateSchedule : IJclSchedule
7442: Function NullStamp : TTimeStamp
7443: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7444: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7445: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7446:
7447: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7448: begin
7449:  AddTypeS('TFunc', 'function(X : Float) : Float;
7450: Function InitGraphics( Width, Height : Integer) : Boolean
7451: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7452: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7453: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7454: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7455: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7456: Procedure SetGraphTitle( Title : String)
7457: Procedure SetOxTitle( Title : String)
7458: Procedure SetOyTitle( Title : String)
7459: Function GetGraphTitle : String
7460: Function GetOxTitle : String
7461: Function GetOyTitle : String
7462: Procedure PlotOxAxis( Canvas : TCanvas)
7463: Procedure PlotOyAxis( Canvas : TCanvas)
7464: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7465: Procedure WriteGraphTitle( Canvas : TCanvas)
7466: Function SetMaxCurv( NCurv : Byte) : Boolean
7467: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7468: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7469: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7470: Procedure SetCurvStep( CurvIndex, Step : Integer)
7471: Function GetMaxCurv : Byte
7472: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7473: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7474: Function GetCurvLegend( CurvIndex : Integer) : String
7475: Function GetCurvStep( CurvIndex : Integer) : Integer
7476: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7477: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7478: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7479: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7480: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7481: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7482: Function Xpixel( X : Float) : Integer
7483: Function Ypixel( Y : Float) : Integer
7484: Function Xuser( X : Integer) : Float
7485: Function Yuser( Y : Integer) : Float
7486: end;
7487:
7488: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7489: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7490: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7491: Procedure FFT_Integer_Cleanup
7492: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7493: //unit uPSI_JclStreams;
7494: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7495: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7496: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7497: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7498:
7499: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7500: begin
7501:  FindClass('TOBJECT'),'EInvalidDest
7502:  FindClass('TOBJECT'),'EFCantMove
7503:  Procedure fmxCopyFile( const FileName, DestName : string)
7504:  Procedure fmxMoveFile( const FileName, DestName : string)
```

```
7505:  Function fmxGetFileSize( const FileName : string) : LongInt
7506:  Function fmxFileDateTime( const FileName : string) : TDateTime
7507:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7508:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7509: end;
7510:
7511: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7512: begin
7513:  SIRegister_IFindFileIterator(CL);
7514:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7515: end;
7516:
7517: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7518: begin
7519:  Function SkipWhite( cp : PChar) : PChar
7520:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7521:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7522:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7523: end;
7524:
7525: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7526: begin
7527:   SIRegister_TStringHashMapTraits(CL);
7528:  Function CaseSensitiveTraits : TStringHashMapTraits
7529:  Function CaseInsensitiveTraits : TStringHashMapTraits
7530:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7531:    +'e; Right : PHashNode; end
7532:   //PHashArray', '^THashArray // will not work
7533:   SIRegister_TStringHashMap(CL);
7534:  THashValue', 'Cardinal
7535:  Function StrHash( const s : string) : THashValue
7536:  Function TextHash( const s : string) : THashValue
7537:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7538:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7539:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7540:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7541:   SIRegister_TCaseSensitiveTraits(CL);
7542:   SIRegister_TCaseInsensitiveTraits(CL);
7543:
7544:
7545: //************************************************************************unit uPSI_umath;
7546:  Function uExpo( X : Float) : Float
7547:  Function uExp2( X : Float) : Float
7548:  Function uExp10( X : Float) : Float
7549:  Function uLog( X : Float) : Float
7550:  Function uLog2( X : Float) : Float
7551:  Function uLog10( X : Float) : Float
7552:  Function uLogA( X, A : Float) : Float
7553:  Function uIntPower( X : Float; N : Integer): Float
7554:  Function uPower( X, Y : Float) : Float
7555:  Function SgnGamma( X : Float) : Integer
7556:  Function Stirling( X : Float) : Float
7557:  Function StirLog( X : Float) : Float
7558:  Function Gamma( X : Float) : Float
7559:  Function LnGamma( X : Float) : Float
7560:  Function DiGamma( X : Float) : Float
7561:  Function TriGamma( X : Float) : Float
7562:  Function IGamma( X : Float) : Float
7563:  Function JGamma( X : Float) : Float
7564:  Function InvGamma( X : Float) : Float
7565:  Function Erf( X : Float) : Float
7566:  Function Erfc( X : Float) : Float
7567: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7568: { Correlation coefficient between samples X and Y }
7569: function DBeta(A, B, X : Float) : Float;
7570: { Density of Beta distribution with parameters A and B }
7571: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7572: Function Beta(X, Y : Float) : Float
7573: Function Binomial( N, K : Integer) : Float
7574: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7575: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7576: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7577: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7578: Function DNorm( X : Float) : Float
7579:
7580: function DGamma(A, B, X : Float) : Float;
7581: { Density of Gamma distribution with parameters A and B }
7582: function DKhi2(Nu : Integer; X : Float) : Float;
7583: { Density of Khi-2 distribution with Nu d.o.f. }
7584: function DStudent(Nu : Integer; X : Float) : Float;
7585: { Density of Student distribution with Nu d.o.f. }
7586: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7587: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7588: function IBeta(A, B, X : Float) : Float;
7589: { Incomplete Beta function}
7590: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7591:
7592: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7593: begin
```

```
7594:  Procedure SetOptAlgo( Algo : TOptAlgo)
7595: procedure SetOptAlgo(Algo : TOptAlgo);
7596: { ----------------------------------------------------------------
7597:   Sets the optimization algorithm according to Algo, which must be
7598:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7599:
7600:  Function GetOptAlgo : TOptAlgo
7601:  Procedure SetMaxParam( N : Byte)
7602:  Function GetMaxParam : Byte
7603:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7604:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7605:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7606:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
      Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7607:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
      MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7608:  Procedure SetMCFile( FileName : String)
7609:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7610:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
      LastPar:Integer;V:TMatrix);
7611: end;
7612:
7613: (*----------------------------------------------------------------------------*)
7614: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7615: begin
7616:  Procedure SaveSimplex( FileName : string)
7617:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7618: end;
7619: (*----------------------------------------------------------------------------*)
7620: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7621: begin
7622:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7623:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7624: end;
7625:
7626: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7627: begin
7628:  Function LTrim( S : String) : String
7629:  Function RTrim( S : String) : String
7630:  Function uTrim( S : String) : String
7631:  Function StrChar( N : Byte; C : Char) : String
7632:  Function RFill( S : String; L : Byte) : String
7633:  Function LFill( S : String; L : Byte) : String
7634:  Function CFill( S : String; L : Byte) : String
7635:  Function Replace( S : String; C1, C2 : Char) : String
7636:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7637:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7638:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7639:  Function FloatStr( X : Float) : String
7640:  Function IntStr( N : LongInt) : String
7641:  Function uCompStr( Z : Complex) : String
7642: end;
7643:
7644: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7645: begin
7646:  Function uSinh( X : Float) : Float
7647:  Function uCosh( X : Float) : Float
7648:  Function uTanh( X : Float) : Float
7649:  Function uArcSinh( X : Float) : Float
7650:  Function uArcCosh( X : Float) : Float
7651:  Function ArcTanh( X : Float) : Float
7652:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7653: end;
7654:
7655: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7656: begin
7657: type RNG_Type =
7658:   (RNG_MWC,      { Multiply-With-Carry }
7659:    RNG_MT,       { Mersenne Twister }
7660:    RNG_UVAG);    { Universal Virtual Array Generator }
7661:  Procedure SetRNG( RNG : RNG_Type)
7662:  Procedure InitGen( Seed : RNG_IntType)
7663:  Procedure SRand( Seed : RNG_IntType)
7664:  Function IRanGen : RNG_IntType
7665:  Function IRanGen31 : RNG_IntType
7666:  Function RanGen1 : Float
7667:  Function RanGen2 : Float
7668:  Function RanGen3 : Float
7669:  Function RanGen53 : Float
7670: end;
7671:
7672: //  Optimization by Simulated Annealing
7673: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7674: begin
7675:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7676:  Procedure SA_CreateLogFile( FileName : String)
7677:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7678: end;
7679:
```

```
7680: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7681: begin
7682:  Procedure InitUVAGbyString( KeyPhrase : string)
7683:  Procedure InitUVAG( Seed : RNG_IntType)
7684:  Function IRanUVAG : RNG_IntType
7685: end;
7686:
7687: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7688: begin
7689:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7690:  Procedure GA_CreateLogFile( LogFileName : String)
7691:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7692: end;
7693:
7694:  TVector', 'array of Float
7695: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7696: begin
7697:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7698:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7699: end;
7700:
7701: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7702: begin
7703:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7704:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7705: end;
7706:
7707: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7708: begin
7709:   FT_Result', 'Integer
7710:   //TDWordptr', '^DWord // will not work
7711:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7712:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7713:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7714:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7715:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7716:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7717:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7718:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7719:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7720:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7721:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7722:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7723:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7724:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7725:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7726:    yte; end
7727: end;
7728:
7729:
7730: //************************************ PaintFX*************************
7731: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7732: begin
7733:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7734:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7735:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7736:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7737:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7738:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7739:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7740:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7741:     Procedure Turn( Src, Dst : TBitmap)
7742:     Procedure TurnRight( Src, Dst : TBitmap)
7743:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7744:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7745:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7746:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7747:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7748:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7749:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7750:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7751:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7752:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7753:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7754:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7755:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7756:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7757:     Procedure Emboss( var Bmp : TBitmap)
7758:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7759:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7760:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7761:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7762:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7763:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7764:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7765:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7766:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7767:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7768:     Procedure SemiOpaque( Src, Dst : TBitmap)
```

```
7769:     Procedure ShadowDownLeft( const Dst : TBitmap)
7770:     Procedure ShadowDownRight( const Dst : TBitmap)
7771:     Procedure ShadowUpLeft( const Dst : TBitmap)
7772:     Procedure ShadowUpRight( const Dst : TBitmap)
7773:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7774:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7775:     Procedure FlipRight( const Dst : TBitmap)
7776:     Procedure FlipDown( const Dst : TBitmap)
7777:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7778:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7779:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7780:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7781:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7782:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7783:     Procedure SmoothResize( var Src, Dst : TBitmap)
7784:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7785:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7786:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7787:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7788:     Procedure GrayScale( const Dst : TBitmap)
7789:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7790:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7791:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7792:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7793:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7794:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7795:     Procedure AntiAlias( const Dst : TBitmap)
7796:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7797:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7798:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7799:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7800:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7801:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7802:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7803:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7804:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7805:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7806:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7807:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7808:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7809:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7810:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7811:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7812:     Procedure Invert( Src : TBitmap)
7813:     Procedure MirrorRight( Src : TBitmap)
7814:     Procedure MirrorDown( Src : TBitmap)
7815:   end;
7816: end;
7817:
7818: (*----------------------------------------------------------------------------*)
7819: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7820: begin
7821:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7822:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7823:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7824:   SIRegister_TJvPaintFX(CL);
7825:  Function SplineFilter( Value : Single) : Single
7826:  Function BellFilter( Value : Single) : Single
7827:  Function TriangleFilter( Value : Single) : Single
7828:  Function BoxFilter( Value : Single) : Single
7829:  Function HermiteFilter( Value : Single) : Single
7830:  Function Lanczos3Filter( Value : Single) : Single
7831:  Function MitchellFilter( Value : Single) : Single
7832: end;
7833:
7834:
7835: (*----------------------------------------------------------------------------*)
7836: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7837: begin
7838:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7839:  TeeMsg_DefaultSeriesName','String 'Series
7840:  TeeMsg_DefaultToolName','String 'ChartTool
7841:  ChartComponentPalette','String 'TeeChart
7842:  TeeMaxLegendColumns',LongInt'( 2);
7843:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7844:  TeeTitleFootDistance,LongInt( 5);
7845:   SIRegister_TCustomChartWall(CL);
7846:   SIRegister_TChartWall(CL);
7847:   SIRegister_TChartLegendGradient(CL);
7848:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7849:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7850:   FindClass('TOBJECT'),'LegendException
7851:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7852:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7853:   FindClass('TOBJECT'),'TCustomChartLegend
7854:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7855:   TLegendSymbolPosition', '( spLeft, spRight )
7856:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7857:   TSymbolCalcHeight', 'Function  : Integer
```

```
7858:   SIRegister_TLegendSymbol(CL);
7859:   SIRegister_TTeeCustomShapePosition(CL);
7860:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7861:   SIRegister_TLegendTitle(CL);
7862:   SIRegister_TLegendItem(CL);
7863:   SIRegister_TLegendItems(CL);
7864:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7865:   FindClass('TOBJECT'),'TCustomChart
7866:   SIRegister_TCustomChartLegend(CL);
7867:   SIRegister_TChartLegend(CL);
7868:   SIRegister_TChartTitle(CL);
7869:   SIRegister_TChartFootTitle(CL);
7870:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7871:    +'eButton; Shift : TShiftState; X, Y : Integer)
7872:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7873:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7874:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7875:    +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7876:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7877:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7878:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7879:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7880:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7881:    +'toMax : Boolean; Min : Double; Max : Double; end
7882:   TAllAxisSavedScales', 'array of TAxisSavedScales
7883:   SIRegister_TChartBackWall(CL);
7884:   SIRegister_TChartRightWall(CL);
7885:   SIRegister_TChartBottomWall(CL);
7886:   SIRegister_TChartLeftWall(CL);
7887:   SIRegister_TChartWalls(CL);
7888:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7889:   SIRegister_TCustomChart(CL);
7890:   SIRegister_TChart(CL);
7891:   SIRegister_TTeeSeriesTypes(CL);
7892:   SIRegister_TTeeToolTypes(CL);
7893:   SIRegister_TTeeDragObject(CL);
7894:   SIRegister_TColorPalettes(CL);
7895:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7896:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7897:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7898:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7899:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7900:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7901:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7902:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7903:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7904:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7905:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7906:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7907:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7908:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7909:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7910:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7911:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7912:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7913:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7914:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7915:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
      R:TRect;ReferenceChart:TCustomChart);
7916:   SIRegister_TChartTheme(CL);
7917:   //TChartThemeClass', 'class of TChartTheme
7918:   //TCanvasClass', 'class of TCanvas3D
7919:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7920:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7921:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7922:  Procedure ShowMessageUser( const S : String)
7923:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7924:  Function HasLabels( ASeries : TChartSeries) : Boolean
7925:  Function HasColors( ASeries : TChartSeries) : Boolean
7926:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7927:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7928: end;
7929:
7930:
7931: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7932: begin
7933:  //'TeeFormBorderStyle',' bsNone);
7934:   SIRegister_TMetafile(CL);
7935:  'TeeDefVerticalMargin','LongInt'( 4);
7936:  'TeeDefHorizMargin','LongInt'( 3);
7937:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7938:  'TeeMsg_TeeHand','String 'crTeeHand
7939:  'TeeNormalPrintDetail','LongInt'( 0);
7940:  'TeeHighPrintDetail','LongInt'( - 100);
7941:  'TeeDefault_PrintMargin','LongInt'( 15);
```

```
7942:  'MaxDefaultColors','LongInt'( 19);
7943:  'TeeTabDelimiter','Char #9);
7944:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7945:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7946:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7947:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7948:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7949:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7950:   SIRegister_TCustomPanelNoCaption(CL);
7951:   FindClass('TOBJECT'),'TCustomTeePanel
7952:   SIRegister_TZoomPanning(CL);
7953:   SIRegister_TTeeEvent(CL);
7954:   //SIRegister_TTeeEventListeners(CL);
7955:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7956:   SIRegister_TTeeMouseEvent(CL);
7957:   SIRegister_TCustomTeePanel(CL);
7958:   //TChartGradient', 'TTeeGradient
7959:   //TChartGradientClass', 'class of TChartGradient
7960:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7961:   SIRegister_TTeeZoomPen(CL);
7962:   SIRegister_TTeeZoomBrush(CL);
7963:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7964:   SIRegister_TTeeZoom(CL);
7965:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7966:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7967:   SIRegister_TBackImage(CL);
7968:   SIRegister_TCustomTeePanelExtended(CL);
7969:   //TChartBrushClass', 'class of TChartBrush
7970:   SIRegister_TTeeCustomShapeBrushPen(CL);
7971:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7972:   TTextFormat', '( ttfNormal, ttfHtml )
7973:   SIRegister_TTeeCustomShape(CL);
7974:   SIRegister_TTeeShape(CL);
7975:   SIRegister_TTeeExportData(CL);
7976:  Function TeeStr( const Num : Integer) : String
7977:  Function DateTimeDefaultFormat( const AStep : Double) : String
7978:  Function TEEDaysInMonth( Year, Month : Word) : Word
7979:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7980:  Function NextDateTimeStep( const AStep : Double) : Double
7981:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7982:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7983:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7984:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7985:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7986:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7987:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7988:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7989:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7990:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7991:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7992:  Function DelphiToLocalFormat( const Format : String) : String
7993:  Function LocalToDelphiFormat( const Format : String) : String
7994:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
7995:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7996:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7997:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
7998:   TTeeSortSwap', 'Procedure ( a, b : Integer)
7999:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
8000:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
8001:  Function TeeExtractField( St : String; Index : Integer) : String;
8002:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
8003:  Function TeeNumFields( St : String) : Integer;
8004:  Function TeeNumFields1( const St, Separator : String) : Integer;
8005:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
8006:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
8007:  // TColorArray', 'array of TColor
8008:  Function GetDefaultColor( const Index : Integer) : TColor
8009:  Procedure SetDefaultColorPalette;
8010:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
8011:  'TeeCheckBoxSize','LongInt'( 11);
8012:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
8013:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
8014:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
8015:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
8016:  Procedure TeeTranslateControl( AControl : TControl);
8017:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
8018:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
8019:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
8020:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
8021:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
8022:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
8023:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
8024:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
8025:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
8026:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
8027:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
8028:  Procedure TeeSaveStringOption( const AKey, Value : String)
8029:  Function TeeDefaultXMLEncoding : String
```

```
8030:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
8031:   TeeWindowHandle', 'Integer
8032:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
8033:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
8034:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
8035: end;
8036:
8037:
8038: using mXBDEUtils
8039: ********************************************************************************
8040:  Procedure SetAlias( aAlias, aDirectory : String)
8041:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
     Desired:Variant;Size:Byte);
8042:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
8043:  Procedure SetBDE( aPath, aNode, aValue : String)
8044:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
8045:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
8046:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
8047:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
8048:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
8049:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8050:
8051:
8052: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
8053: begin
8054: AddClassN(FindClass('TOBJECT'),'EDateTime
8055: Function DatePart( const D : TDateTime) : Integer
8056: Function TimePart( const D : TDateTime) : Double
8057: Function Century( const D : TDateTime) : Word
8058: Function Year( const D : TDateTime) : Word
8059: Function Month( const D : TDateTime) : Word
8060: Function Day( const D : TDateTime) : Word
8061: Function Hour( const D : TDateTime) : Word
8062: Function Minute( const D : TDateTime) : Word
8063: Function Second( const D : TDateTime) : Word
8064: Function Millisecond( const D : TDateTime) : Word
8065: ('OneDay','Extended').setExtended( 1.0);
8066: ('OneHour','Extended').SetExtended( OneDay / 24);
8067: ('OneMinute','Extended').SetExtended( OneHour / 60);
8068: ('OneSecond','Extended').SetExtended( OneMinute / 60);
8069: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8070: ('OneWeek','Extended').SetExtended( OneDay * 7);
8071: ('HoursPerDay','Extended').SetExtended( 24);
8072: ('MinutesPerHour','Extended').SetExtended( 60);
8073: ('SecondsPerMinute','Extended').SetExtended( 60);
8074: Procedure SetYear( var D : TDateTime; const Year : Word)
8075: Procedure SetMonth( var D : TDateTime; const Month : Word)
8076: Procedure SetDay( var D : TDateTime; const Day : Word)
8077: Procedure SetHour( var D : TDateTime; const Hour : Word)
8078: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8079: Procedure SetSecond( var D : TDateTime; const Second : Word)
8080: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8081: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8082: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8083: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8084: Function IsAM( const D : TDateTime) : Boolean
8085: Function IsPM( const D : TDateTime) : Boolean
8086: Function IsMidnight( const D : TDateTime) : Boolean
8087: Function IsNoon( const D : TDateTime) : Boolean
8088: Function IsSunday( const D : TDateTime) : Boolean
8089: Function IsMonday( const D : TDateTime) : Boolean
8090: Function IsTuesday( const D : TDateTime) : Boolean
8091: Function IsWedneday( const D : TDateTime) : Boolean
8092: Function IsThursday( const D : TDateTime) : Boolean
8093: Function IsFriday( const D : TDateTime) : Boolean
8094: Function IsSaturday( const D : TDateTime) : Boolean
8095: Function IsWeekend( const D : TDateTime) : Boolean
8096: Function Noon( const D : TDateTime) : TDateTime
8097: Function Midnight( const D : TDateTime) : TDateTime
8098: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8099: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8100: Function NextWorkday( const D : TDateTime) : TDateTime
8101: Function PreviousWorkday( const D : TDateTime) : TDateTime
8102: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8103: Function LastDayOfYear( const D : TDateTime) : TDateTime
8104: Function EasterSunday( const Year : Word) : TDateTime
8105: Function GoodFriday( const Year : Word) : TDateTime
8106: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8107: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8108: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8109: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8110: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8111: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8112: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8113: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8114: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8115: Function DayOfYear( const D : TDateTime) : Integer
8116: Function DaysInMonth( const Ye, Mo : Word) : Integer
8117: Function DaysInMonth( const D : TDateTime) : Integer
```

```
8118: Function DaysInYear( const Ye : Word) : Integer
8119: Function DaysInYearDate( const D : TDateTime) : Integer
8120: Function WeekNumber( const D : TDateTime) : Integer
8121: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8122: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8123: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8124: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8125: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8126: Function DiffHours( const D1, D2 : TDateTime) : Integer
8127: Function DiffDays( const D1, D2 : TDateTime) : Integer
8128: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8129: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8130: Function DiffYears( const D1, D2 : TDateTime) : Integer
8131: Function GMTBias : Integer
8132: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8133: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8134: Function NowAsGMTTime : TDateTime
8135: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8136: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8137: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8138: Function DateTimeToANSI( const D : TDateTime) : Integer
8139: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8140: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8141: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8142: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8143: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8144: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8145: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8146: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8147: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8148: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8149: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8150: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8151: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8152: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8153: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8154: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8155: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8156: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8157: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8158: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8159: Function EnglishShortMonthA( const S : AnsiString) : Integer
8160: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8161: Function EnglishLongMonthA( const S : AnsiString) : Integer
8162: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8163: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8164: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8165: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8166: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8167: Function RFCMonthA( const S : AnsiString) : Word
8168: Function RFCMonthU( const S : UnicodeString) : Word
8169: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8170: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8171: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8172: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8173: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8174: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8175: Function NowAsRFCDateTimeA : AnsiString
8176: Function NowAsRFCDateTimeU : UnicodeString
8177: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8178: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8179: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8180: Function TimePeriodStr( const D : TDateTime) : AnsiString
8181: Procedure SelfTest
8182: end;
8183: //*****************************************CFileUtils
8184: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8185: Function PathHasDriveLetter( const Path : String) : Boolean
8186: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8187: Function PathIsDriveLetter( const Path : String) : Boolean
8188: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8189: Function PathIsDriveRoot( const Path : String) : Boolean
8190: Function PathIsRootA( const Path : AnsiString) : Boolean
8191: Function PathIsRoot( const Path : String) : Boolean
8192: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8193: Function PathIsUNCPath( const Path : String) : Boolean
8194: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8195: Function PathIsAbsolute( const Path : String) : Boolean
8196: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8197: Function PathIsDirectory( const Path : String) : Boolean
8198: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8199: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8200: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8201: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8202: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8203: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8204: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8205: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8206: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
```

```
8207: Function PathCanonical( const Path : String; const PathSep : Char) : String
8208: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8209: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8210: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8211: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8212: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8213: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8214: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8215: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8216: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8217: Function FileNameValid( const FileName : String) : String
8218: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8219: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8220: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8221: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8222: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8223: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8224: Procedure CCopyFile( const FileName, DestName : String)
8225: Procedure CMoveFile( const FileName, DestName : String)
8226: Function CDeleteFiles( const FileMask : String) : Boolean
8227: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8228: Procedure FileCloseEx( const FileHandle : TFileHandle)
8229: Function FileExistsA( const FileName : AnsiString) : Boolean
8230:  Function CFileExists( const FileName : String) : Boolean
8231:  Function CFileGetSize( const FileName : String) : Int64
8232: Function FileGetDateTime( const FileName : String) : TDateTime
8233: Function FileGetDateTime2( const FileName : String) : TDateTime
8234: Function FileIsReadOnly( const FileName : String) : Boolean
8235: Procedure FileDeleteEx( const FileName : String)
8236: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8237: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
      : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8238: Function DirectoryEntryExists( const Name : String) : Boolean
8239: Function DirectoryEntrySize( const Name : String) : Int64
8240:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8241: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8242: Procedure CDirectoryCreate( const DirectoryName : String)
8243: Function GetFirstFileNameMatching( const FileMask : String) : String
8244: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8245: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8246: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8247:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8248:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8249: Function DriveIsValid( const Drive : Char) : Boolean
8250: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8251: Function DriveFreeSpace( const Path : AnsiString) : Int64
8252:
8253: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8254: begin
8255:  AddClassN(FindClass('TOBJECT'),'ETimers
8256:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8257: Function TickDelta( const D1, D2 : LongWord) : Integer
8258: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8259:   AddTypeS('THPTimer', 'Int64
8260: Procedure StartTimer( var Timer : THPTimer)
8261: Procedure StopTimer( var Timer : THPTimer)
8262: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8263: Procedure InitStoppedTimer( var Timer : THPTimer)
8264: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8265: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8266: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8267: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8268: Function GetHighPrecisionFrequency : Int64
8269: Function GetHighPrecisionTimerOverhead : Int64
8270: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8271: Procedure SelfTestCTimer
8272: end;
8273:
8274: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8275: begin
8276:  Function RandomSeed : LongWord
8277:  Procedure AddEntropy( const Value : LongWord)
8278:  Function RandomUniform : LongWord;
8279:  Function RandomUniform1( const N : Integer) : Integer;
8280:  Function RandomBoolean : Boolean
8281:  Function RandomByte : Byte
8282:  Function RandomByteNonZero : Byte
8283:  Function RandomWord : Word
8284:  Function RandomInt64 : Int64;
8285:  Function RandomInt641( const N : Int64) : Int64;
8286:  Function RandomHex( const Digits : Integer) : String
8287:  Function RandomFloat : Extended
8288:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8289:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8290:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8291:  Function mwcRandomLongWord : LongWord
8292:  Function urnRandomLongWord : LongWord
8293:  Function moaRandomFloat : Extended
8294:  Function mwcRandomFloat : Extended
```

```
8295:  Function RandomNormalF : Extended
8296:  Procedure SelfTestCRandom
8297: end;
8298:
8299: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8300: begin
8301:  // PIntArray', '^TIntArray // will not work
8302:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8303:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8304:  Function synMax( x, y : integer) : integer
8305:  Function synMin( x, y : integer) : integer
8306:  Function synMinMax( x, mi, ma : integer) : integer
8307:  Procedure synSwapInt( var l, r : integer)
8308:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8309:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8310:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8311:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8312:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8313:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8314:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8315:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8316:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8317:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8318:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8319:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8320:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8321:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8322:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8323:  ('C3_NONSPACING','LongInt'( 1);
8324:  'C3_DIACRITIC','LongInt'( 2);
8325:  'C3_VOWELMARK','LongInt'( 4);
8326:  ('C3_SYMBOL','LongInt'( 8);
8327:  ('C3_KATAKANA','LongWord( $0010);
8328:  ('C3_HIRAGANA','LongWord( $0020);
8329:  ('C3_HALFWIDTH','LongWord( $0040);
8330:  ('C3_FULLWIDTH','LongWord( $0080);
8331:  ('C3_IDEOGRAPH','LongWord( $0100);
8332:  ('C3_KASHIDA','LongWord( $0200);
8333:  ('C3_LEXICAL','LongWord( $0400);
8334:  ('C3_ALPHA','LongWord( $8000);
8335:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8336:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8337:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8338:  Function synIsStringType( Value : Word) : TStringType
8339:  Function synGetEOL( Line : PChar) : PChar
8340:  Function synEncodeString( s : string) : string
8341:  Function synDecodeString( s : string) : string
8342:  Procedure synFreeAndNil( var Obj: TObject)
8343:  Procedure synAssert( Expr : Boolean)
8344:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8345:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8346:   TReplaceFlags', 'set of TReplaceFlag )
8347:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8348:  Function synGetRValue( RGBValue : TColor) : byte
8349:  Function synGetGValue( RGBValue : TColor) : byte
8350:  Function synGetBValue( RGBValue : TColor) : byte
8351:  Function synRGB( r, g, b : Byte) : Cardinal
8352: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8353:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8354:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8355:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8356:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8357: end;
8358:
8359:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8360:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8361:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8362:
8363: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8364: begin
8365:  Function STimeZoneBias : integer
8366:  Function TimeZone : string
8367:  Function Rfc822DateTime( t : TDateTime) : string
8368:  Function CDateTime( t : TDateTime) : string
8369:  Function SimpleDateTime( t : TDateTime) : string
8370:  Function AnsiCDateTime( t : TDateTime) : string
8371:  Function GetMonthNumber( Value : String) : integer
8372:  Function GetTimeFromStr( Value : string) : TDateTime
8373:  Function GetDateMDYFromStr( Value : string) : TDateTime
8374:  Function DecodeRfcDateTime( Value : string) : TDateTime
8375:  Function GetUTTime : TDateTime
8376:  Function SetUTTime( Newdt : TDateTime) : Boolean
8377:  Function SGetTick : LongWord
8378:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8379:  Function CodeInt( Value : Word) : Ansistring
8380:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8381:  Function CodeLongInt( Value : LongInt) : Ansistring
```

```
8382:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8383:  Function DumpStr( const Buffer : Ansistring) : string
8384:  Function DumpExStr( const Buffer : Ansistring) : string
8385:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8386:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8387:  Function TrimSPLeft( const S : string) : string
8388:  Function TrimSPRight( const S : string) : string
8389:  Function TrimSP( const S : string) : string
8390:  Function SeparateLeft( const Value, Delimiter : string) : string
8391:  Function SeparateRight( const Value, Delimiter : string) : string
8392:  Function SGetParameter( const Value, Parameter : string) : string
8393:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8394:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8395:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8396:  Function GetEmailAddr( const Value : string) : string
8397:  Function GetEmailDesc( Value : string) : string
8398:  Function CStrToHex( const Value : Ansistring) : string
8399:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8400:  Function CBinToInt( const Value : string) : Integer
8401:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8402:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8403:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8404:  Function CRPos( const Sub, Value : String) : Integer
8405:  Function FetchBin( var Value : string; const Delimiter : string) : string
8406:  Function CFetch( var Value : string; const Delimiter : string) : string
8407:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8408:  Function IsBinaryString( const Value : AnsiString) : Boolean
8409:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8410:  Procedure StringsTrim( const value : TStrings)
8411:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8412:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8413:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8414:  Function CCountOfChar( const Value : string; aChr : char) : integer
8415:  Function UnquoteStr( const Value : string; Quote : Char) : string
8416:  Function QuoteStr( const Value : string; Quote : Char) : string
8417:  Procedure HeadersToList( const Value : TStrings)
8418:  Procedure ListToHeaders( const Value : TStrings)
8419:  Function SwapBytes( Value : integer) : integer
8420:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8421:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8422:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8423:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8424:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8425:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8426: end;
8427:
8428: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8429: begin
8430:  ('CrcBufSize','LongInt'( 2048);
8431:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8432:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8433:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8434:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8435:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8436:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8437:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8438:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8439:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8440:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8441:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8442:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8443:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8444:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8445:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8446: end;
8447:
8448: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8449: begin
8450:  function CreateOleObject(const ClassName: String): IDispatch;
8451:  function GetActiveOleObject(const ClassName: String): IDispatch;
8452:  function ProgIDToClassID(const ProgID: string): TGUID;
8453:  function ClassIDToProgID(const ClassID: TGUID): string;
8454:  function CreateClassID: string;
8455:  function CreateGUIDString: string;
8456:  function CreateGUIDID: string;
8457:  procedure OleError(ErrorCode: longint)
8458:  procedure OleCheck(Result: HResult);
8459: end;
8460:
8461:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8462:  Function xGetActiveOleObject( const ClassName : string) : Variant
8463:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8464:  Function DllCanUnloadNow : HResult
8465:  Function DllRegisterServer : HResult
8466:  Function DllUnregisterServer : HResult
8467:  Function VarFromInterface( Unknown : IUnknown) : Variant
8468:  Function VarToInterface( const V : Variant) : IDispatch
8469:  Function VarToAutoObject( const V : Variant) : TAutoObject
8470:  //Procedure
DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
```

```
8471:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8472:  Procedure OleError( ErrorCode : HResult)
8473:  Procedure OleCheck( Result : HResult)
8474:  Function StringToClassID( const S : string) : TCLSID
8475:  Function ClassIDToString( const ClassID : TCLSID) : string
8476:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8477:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8478:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8479:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8480:  Function xGUIDToString( const ClassID : TGUID) : string
8481:  Function xStringToGUID( const S : string) : TGUID
8482:  Function xGetModuleName( Module : HMODULE) : string
8483:  Function xAcquireExceptionObject : TObject
8484:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8485:  Function xUtf8Encode( const WS : WideString) : UTF8String
8486:  Function xUtf8Decode( const S : UTF8String) : WideString
8487:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8488:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8489:  Function XRTLHandleCOMException : HResult
8490:  Procedure XRTLCheckArgument( Flag : Boolean)
8491:  //Procedure XRTLCheckOutArgument( out Arg)
8492:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8493:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8494:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8495:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8496:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8497:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8498: function   XRTLDefaultCategoryManager: IUnknown;
8499: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8500: // ICatRegister helper functions
8501: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8502:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8503:                                        const CategoryManager: IUnknown = nil): HResult;
8504: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8505:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8506:                                        const CategoryManager: IUnknown = nil): HResult;
8507: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8508:                                        const CategoryManager: IUnknown = nil): HResult;
8509: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8510:                                          const CategoryManager: IUnknown = nil): HResult;
8511: // ICatInformation helper functions
8512: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8513:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8514:                                       const CategoryManager: IUnknown = nil): HResult;
8515: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8516:                                const CategoryManager: IUnknown = nil): HResult;
8517: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8518:                                     const CategoryManager: IUnknown = nil): HResult;
8519: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8520:                                      const CategoryManager: IUnknown = nil): HResult;
8521: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8522:                    const ADelete: Boolean = True): WideString;
8523: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8524: Function XRTLGetVariantAsString( const Value : Variant) : string
8525: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8526: Function XRTLGetTimeZones : TXRTLTimeZones
8527: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8528: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8529: Function GMTNow : TDateTime
8530: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8531: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8532: Procedure XRTLNotImplemented
8533: Procedure XRTLRaiseError( E : Exception)
8534: Procedure XRTLRaise( E : Exception)');
8535: Procedure XRaise( E : Exception)');
8536: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8537:
8538:
8539: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8540: begin
8541:   SIRegister_IXRTLValue(CL);
8542:   SIRegister_TXRTLValue(CL);
8543:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8544:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8545: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8546: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8547: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8548: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8549: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8550: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8551: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8552: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8553: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8554: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8555: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8556: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8557: Function XRTLValue3( const AValue : Single) : IXRTLValue;
```

```
8558: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8559: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8560: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8561: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8562: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8563: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8564: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8565: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8566: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8567: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8568: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8569: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8570: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8571: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8572:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8573: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8574: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8575: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8576: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8577: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8578: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8579: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8580: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8581: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8582: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8583: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8584: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8585: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8586: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8587: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8588: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8589: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8590: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8591: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8592: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8593: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8594: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8595: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8596: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8597: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8598: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8599: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8600: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8601: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8602: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8603: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8604: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8605: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8606: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8607: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8608: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8609: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8610: end;
8611:
8612: //*****************************unit uPSI_GR32;****************************************
8613:
8614:  Function Color32( WinColor : TColor) : TColor32;
8615:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8616:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8617:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8618:  Function WinColor( Color32 : TColor32) : TColor
8619:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8620:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8621:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8622:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8623:  Function RedComponent( Color32 : TColor32) : Integer
8624:  Function GreenComponent( Color32 : TColor32) : Integer
8625:  Function BlueComponent( Color32 : TColor32) : Integer
8626:  Function AlphaComponent( Color32 : TColor32) : Integer
8627:  Function Intensity( Color32 : TColor32) : Integer
8628:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8629:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8630:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8631:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8632:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8633:  Function WinPalette( const P : TPalette32) : HPALETTE
8634:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8635:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8636:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8637:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8638:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8639:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8640:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8641:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8642:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8643:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8644:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8645:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
```

```
8646:  Function FixedRect1( const ARect : TRect) : TRect;
8647:  Function FixedRect2( const FR : TFloatRect) : TRect;
8648:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8649:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8650:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8651:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8652:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8653:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8654:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8655:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8656:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8657:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8658:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8659:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8660:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8661:  Function IsRectEmpty( const R : TRect) : Boolean;
8662:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8663:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8664:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8665:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8666:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8667:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8668:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8669:  Function MessageBeep( uType : UINT) : BOOL
8670:  Function ShowCursor( bShow : BOOL) : Integer
8671:  Function SetCursorPos( X, Y : Integer) : BOOL
8672:  Function SetCursor( hCursor : HICON) : HCURSOR
8673:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8674:  //Function ClipCursor( lpRect : PRect) : BOOL
8675:  Function GetClipCursor( var lpRect : TRect) : BOOL
8676:  Function GetCursor : HCURSOR
8677:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8678:  Function GetCaretBlinkTime : UINT
8679:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8680:  Function DestroyCaret : BOOL
8681:  Function HideCaret( hWnd : HWND) : BOOL
8682:  Function ShowCaret( hWnd : HWND) : BOOL
8683:  Function SetCaretPos( X, Y : Integer) : BOOL
8684:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8685:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8686:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8687:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8688:  Function WindowFromPoint( Point : TPoint) : HWND
8689:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8690:
8691:
8692: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8693: begin
8694:  Function FixedFloor( A : TFixed) : Integer
8695:  Function FixedCeil( A : TFixed) : Integer
8696:  Function FixedMul( A, B : TFixed) : TFixed
8697:  Function FixedDiv( A, B : TFixed) : TFixed
8698:  Function OneOver( Value : TFixed) : TFixed
8699:  Function FixedRound( A : TFixed) : Integer
8700:  Function FixedSqr( Value : TFixed) : TFixed
8701:  Function FixedSqrtLP( Value : TFixed) : TFixed
8702:  Function FixedSqrtHP( Value : TFixed) : TFixed
8703:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8704:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8705:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8706:  Function GRHypot( const X, Y : TFloat) : TFloat;
8707:  Function Hypot1( const X, Y : Integer) : Integer;
8708:  Function FastSqrt( const Value : TFloat) : TFloat
8709:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8710:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8711:  Function FastInvSqrt( const Value : Single) : Single;
8712:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8713:  Function GRIsPowerOf2( Value : Integer) : Boolean
8714:  Function PrevPowerOf2( Value : Integer) : Integer
8715:  Function NextPowerOf2( Value : Integer) : Integer
8716:  Function Average( A, B : Integer) : Integer
8717:  Function GRSign( Value : Integer) : Integer
8718:  Function FloatMod( x, y : Double) : Double
8719: end;
8720:
8721: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8722: begin
8723:  Function Clamp( const Value : Integer) : Integer;
8724:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8725:  Function StackAlloc( Size : Integer) : Pointer
8726:  Procedure StackFree( P : Pointer)
8727:  Procedure Swap( var A, B : Pointer);
8728:  Procedure Swap1( var A, B : Integer);
8729:  Procedure Swap2( var A, B : TFixed);
8730:  Procedure Swap3( var A, B : TColor32);
8731:  Procedure TestSwap( var A, B : Integer);
8732:  Procedure TestSwap1( var A, B : TFixed);
8733:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8734:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
```

```
8735:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8736:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8737:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8738:  Function GRMin( const A, B, C : Integer) : Integer;
8739:  Function GRMax( const A, B, C : Integer) : Integer;
8740:  Function Clamp( Value, Max : Integer) : Integer;
8741:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8742:  Function Wrap( Value, Max : Integer) : Integer;
8743:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8744:  Function Wrap3( Value, Max : Single) : Single;;
8745:  Function WrapPow2( Value, Max : Integer) : Integer;
8746:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8747:  Function Mirror( Value, Max : Integer) : Integer;
8748:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8749:  Function MirrorPow2( Value, Max : Integer) : Integer;
8750:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8751:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8752:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8753:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8754:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8755:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8756:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8757:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8758:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8759:  Function Div255( Value : Cardinal) : Cardinal
8760:  Function SAR_4( Value : Integer) : Integer
8761:  Function SAR_8( Value : Integer) : Integer
8762:  Function SAR_9( Value : Integer) : Integer
8763:  Function SAR_11( Value : Integer) : Integer
8764:  Function SAR_12( Value : Integer) : Integer
8765:  Function SAR_13( Value : Integer) : Integer
8766:  Function SAR_14( Value : Integer) : Integer
8767:  Function SAR_15( Value : Integer) : Integer
8768:  Function SAR_16( Value : Integer) : Integer
8769:  Function ColorSwap( WinColor : TColor) : TColor32
8770: end;
8771:
8772: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8773: begin
8774:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8775:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8776:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8777:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8778:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8779:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8780:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8781:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8782:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8783:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8784:  Function CreateBitmask( Components : TColor32Components) : TColor32
8785:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8786:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8787:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8788: end;
8789:
8790:
8791: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8792: begin
8793:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8794:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8795:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8796:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8797:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8798:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8799:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8800:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8801:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8802:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8803:    //+'tedRangeBuffer; MoreData : Boolean; end
8804:  Function NtfsSetSparse( const FileName : string) : Boolean
8805:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8806:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8807:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8808:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8809:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8810:  Function NtfsGetSparse( const FileName : string) : Boolean
8811:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8812:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8813:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8814:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8815:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8816:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8817:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8818:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
```

```
8819:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8820:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8821:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8822:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8823:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8824:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8825:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8826:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8827:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8828:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8829:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8830:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8831:   AddTypeS('TStreamIds', 'set of TStreamId
8832:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8833:    +': ___Pointer; StreamIds : TStreamIds; end
8834:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8835:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8836:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8837:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8838:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8839:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8840:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8841:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8842:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8843:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8844:  Function JclAppInstances : TJclAppInstances;
8845:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8846:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8847:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8848:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8849:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8850:
8851:
8852: (*----------------------------------------------------------------------------*)
8853: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8854: begin
8855:   FindClass('TOBJECT'),'EJclGraphicsError
8856:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8857:   TDynPointArray', 'array of TPoint
8858:   TDynDynPointArrayArray', 'array of TDynPointArray
8859:   TPointF', 'record X : Single; Y : Single; end
8860:   TDynPointArrayF', 'array of TPointF
8861:   TDrawMode2', '( dmOpaque, dmBlend )
8862:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8863:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8864:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8865:   TMatrix3d', 'record array[0..2,0..2] of extended end
8866:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8867:   TScanLine', 'array of Integer
8868:   TScanLines', 'array of TScanLine
8869:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8870:   TGradientDirection', '( gdVertical, gdHorizontal )
8871:   TPolyFillMode', '( fmAlternate, fmWinding )
8872:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8873:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8874:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8875:   SIRegister_TJclDesktopCanvas(CL);
8876:   FindClass('TOBJECT'),'TJclRegion
8877:   SIRegister_TJclRegionInfo(CL);
8878:   SIRegister_TJclRegion(CL);
8879:   SIRegister_TJclThreadPersistent(CL);
8880:   SIRegister_TJclCustomMap(CL);
8881:   SIRegister_TJclBitmap32(CL);
8882:   SIRegister_TJclByteMap(CL);
8883:   SIRegister_TJclTransformation(CL);
8884:   SIRegister_TJclLinearTransformation(CL);
8885:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8886:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8887:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8888:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8889:  Procedure BitmapToJPeg( const FileName : string)
8890:  Procedure JPegToBitmap( const FileName : string)
8891:  Function ExtractIconCount( const FileName : string) : Integer
8892:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8893:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8894:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8895:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8896:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8897:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8898:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8899:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8900:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8901:  Procedure ScreenShot1( bm : TBitmap);
```

```
8902:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8903:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8904:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8905:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8906:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8907:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8908:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8909:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8910:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8911:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8912:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8913:  Procedure Invert( Dst, Src : TJclBitmap32)
8914:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8915:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8916:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8917:  Procedure SetGamma( Gamma : Single)
8918: end;
8919:
8920: (*----------------------------------------------------------------------------*)
8921: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8922: begin
8923:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8924:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8925:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8926:  Function LockedDec( var Target : Integer) : Integer
8927:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8928:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8929:  Function LockedExchangeDec( var Target : Integer) : Integer
8930:  Function LockedExchangeInc( var Target : Integer) : Integer
8931:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8932:  Function LockedInc( var Target : Integer) : Integer
8933:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8934:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8935:   SIRegister_TJclDispatcherObject(CL);
8936:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8937:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8938:   SIRegister_TJclCriticalSection(CL);
8939:   SIRegister_TJclCriticalSectionEx(CL);
8940:   SIRegister_TJclEvent(CL);
8941:   SIRegister_TJclWaitableTimer(CL);
8942:   SIRegister_TJclSemaphore(CL);
8943:   SIRegister_TJclMutex(CL);
8944:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8945:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8946:   SIRegister_TJclOptex(CL);
8947:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8948:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8949:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8950:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8951:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8952:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8953:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8954:   PMeteredSection', '^TMeteredSection // will not work
8955:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8956:   SIRegister_TJclMeteredSection(CL);
8957:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8958:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8959:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8960:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8961:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8962:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8963:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8964:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8965:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8966:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8967:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8968:   FindClass('TOBJECT'),'EJclCriticalSectionError
8969:   FindClass('TOBJECT'),'EJclEventError
8970:   FindClass('TOBJECT'),'EJclWaitableTimerError
8971:   FindClass('TOBJECT'),'EJclSemaphoreError
8972:   FindClass('TOBJECT'),'EJclMutexError
8973:   FindClass('TOBJECT'),'EJclMeteredSectionError
8974: end;
8975:
8976:
8977: //*************************unit uPSI_mORMotReport;
8978: Procedure SetCurrentPrinterAsDefault
8979: Function CurrentPrinterName : string
8980: Function mCurrentPrinterPaperSize : string
8981: Procedure UseDefaultPrinter
8982:
8983: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
8984: begin
8985:   with FindClass('TOBJECT'), 'TStream') do begin
8986:     IsAbstract := True;
8987:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8988:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
```

```
8989:     function Read(Buffer:String;Count:LongInt):LongInt
8990:     function Write(Buffer:String;Count:LongInt):LongInt
8991:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
8992:     function WriteString(Buffer:String;Count:LongInt):LongInt
8993:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8994:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
8995:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8996:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8997:
8998:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8999:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
9000:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
9001:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
9002:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
9003:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
9004:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
9005:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
9006:
9007:     function Seek(Offset:LongInt;Origin:Word):LongInt
9008:     procedure ReadBuffer(Buffer:String;Count:LongInt)
9009:     procedure WriteBuffer(Buffer:String;Count:LongInt)
9010:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
9011:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
9012:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
9013:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
9014:
9015:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
9016:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
9017:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
9018:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
9019:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9020:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9021:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9022:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
9023:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
9024:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
9025:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
9026:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
9027:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
9028:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
9029:
9030:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
9031:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
9032:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
9033:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
9034:     //READBUFFERAC
9035:     function InstanceSize: Longint
9036:     Procedure FixupResourceHeader( FixupInfo : Integer)
9037:     Procedure ReadResHeader
9038:
9039:     {$IFDEF DELPHI4UP}
9040:     function CopyFrom(Source:TStream;Count:Int64):LongInt
9041:     {$ELSE}
9042:     function CopyFrom(Source:TStream;Count:Integer):LongInt
9043:     {$ENDIF}
9044:     RegisterProperty('Position', 'LongInt', iptrw);
9045:     RegisterProperty('Size', 'LongInt', iptrw);
9046:   end;
9047: end;
9048:
9049:
9050: { ***************************************************************
9051:   Unit DMATH - Interface for DMATH.DLL
9052:   *************************************************************** }
9053: // see more docs/dmath_manual.pdf
9054:
9055: Function InitEval : Integer
9056: Procedure SetVariable( VarName : Char; Value : Float)
9057: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9058: Function Eval( ExpressionString : String) : Float
9059:
9060: unit dmath; //types are in built, others are external in DLL
9061: interface
9062: {$IFDEF DELPHI}
9063: uses
9064:   StdCtrls, Graphics;
9065: {$ENDIF}
9066: { ---------------------------------------------------------------
9067:   Types and constants
9068:   --------------------------------------------------------------- }
9069: {$i types.inc}
9070: { ---------------------------------------------------------------
9071:   Error handling
9072:   --------------------------------------------------------------- }
9073: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9074: { Sets the error code }
9075: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9076: { Sets error code and default function value }
9077: function MathErr : Integer; external 'dmath';
```

```
9078: { Returns the error code }
9079: { ---------------------------------------------------------------
9080:   Dynamic arrays
9081:   --------------------------------------------------------------- }
9082: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9083: { Sets the auto-initialization of arrays }
9084: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9085: { Creates floating point vector V[0..Ub] }
9086: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9087: { Creates integer vector V[0..Ub] }
9088: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9089: { Creates complex vector V[0..Ub] }
9090: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9091: { Creates boolean vector V[0..Ub] }
9092: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9093: { Creates string vector V[0..Ub] }
9094: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9095: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9096: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9097: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9098: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9099: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9100: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9101: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9102: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9103: { Creates string matrix A[0..Ub1, 0..Ub2] }
9104: { ---------------------------------------------------------------
9105:   Minimum, maximum, sign and exchange
9106:   --------------------------------------------------------------- }
9107: function FMin(X, Y : Float) : Float; external 'dmath';
9108: { Minimum of 2 reals }
9109: function FMax(X, Y : Float) : Float; external 'dmath';
9110: { Maximum of 2 reals }
9111: function IMin(X, Y : Integer) : Integer; external 'dmath';
9112: { Minimum of 2 integers }
9113: function IMax(X, Y : Integer) : Integer; external 'dmath';
9114: { Maximum of 2 integers }
9115: function Sgn(X : Float) : Integer; external 'dmath';
9116: { Sign (returns 1 if X = 0) }
9117: function Sgn0(X : Float) : Integer; external 'dmath';
9118: { Sign (returns 0 if X = 0) }
9119: function DSgn(A, B : Float) : Float; external 'dmath';
9120: { Sgn(B) * |A| }
9121: procedure FSwap(var X, Y : Float); external 'dmath';
9122: { Exchange 2 reals }
9123: procedure ISwap(var X, Y : Integer); external 'dmath';
9124: { Exchange 2 integers }
9125: { ---------------------------------------------------------------
9126:   Rounding functions
9127:   --------------------------------------------------------------- }
9128: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9129: { Rounds X to N decimal places }
9130: function Ceil(X : Float) : Integer; external 'dmath';
9131: { Ceiling function }
9132: function Floor(X : Float) : Integer; external 'dmath';
9133: { Floor function }
9134: { ---------------------------------------------------------------
9135:   Logarithms, exponentials and power
9136:   --------------------------------------------------------------- }
9137: function Expo(X : Float) : Float; external 'dmath';
9138: { Exponential }
9139: function Exp2(X : Float) : Float; external 'dmath';
9140: { 2^X }
9141: function Exp10(X : Float) : Float; external 'dmath';
9142: { 10^X }
9143: function Log(X : Float) : Float; external 'dmath';
9144: { Natural log }
9145: function Log2(X : Float) : Float; external 'dmath';
9146: { Log, base 2 }
9147: function Log10(X : Float) : Float; external 'dmath';
9148: { Decimal log }
9149: function LogA(X, A : Float) : Float; external 'dmath';
9150: { Log, base A }
9151: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9152: { X^N }
9153: function Power(X, Y : Float) : Float; external 'dmath';
9154: { X^Y, X >= 0 }
9155: { ---------------------------------------------------------------
9156:   Trigonometric functions
9157:   --------------------------------------------------------------- }
9158: function Pythag(X, Y : Float) : Float; external 'dmath';
9159: { Sqrt(X^2 + Y^2) }
9160: function FixAngle(Theta : Float) : Float; external 'dmath';
9161: { Set Theta in -Pi..Pi }
9162: function Tan(X : Float) : Float; external 'dmath';
9163: { Tangent }
9164: function ArcSin(X : Float) : Float; external 'dmath';
9165: { Arc sinus }
9166: function ArcCos(X : Float) : Float; external 'dmath';
```

```
9167: { Arc cosinus }
9168: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9169: { Angle (Ox, OM) with M(X,Y) }
9170: { ------------------------------------------------------------
9171:   Hyperbolic functions
9172:   ------------------------------------------------------------ }
9173: function Sinh(X : Float) : Float; external 'dmath';
9174: { Hyperbolic sine }
9175: function Cosh(X : Float) : Float; external 'dmath';
9176: { Hyperbolic cosine }
9177: function Tanh(X : Float) : Float; external 'dmath';
9178: { Hyperbolic tangent }
9179: function ArcSinh(X : Float) : Float; external 'dmath';
9180: { Inverse hyperbolic sine }
9181: function ArcCosh(X : Float) : Float; external 'dmath';
9182: { Inverse hyperbolic cosine }
9183: function ArcTanh(X : Float) : Float; external 'dmath';
9184: { Inverse hyperbolic tangent }
9185: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9186: { Sinh & Cosh }
9187: { ------------------------------------------------------------
9188:   Gamma function and related functions
9189:   ------------------------------------------------------------ }
9190: function Fact(N : Integer) : Float; external 'dmath';
9191: { Factorial }
9192: function SgnGamma(X : Float) : Integer; external 'dmath';
9193: { Sign of Gamma function }
9194: function Gamma(X : Float) : Float; external 'dmath';
9195: { Gamma function }
9196: function LnGamma(X : Float) : Float; external 'dmath';
9197: { Logarithm of Gamma function }
9198: function Stirling(X : Float) : Float; external 'dmath';
9199: { Stirling's formula for the Gamma function }
9200: function StirLog(X : Float) : Float; external 'dmath';
9201: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9202: function DiGamma(X : Float ) : Float; external 'dmath';
9203: { Digamma function }
9204: function TriGamma(X : Float ) : Float; external 'dmath';
9205: { Trigamma function }
9206: function IGamma(A, X : Float) : Float; external 'dmath';
9207: { Incomplete Gamma function}
9208: function JGamma(A, X : Float) : Float; external 'dmath';
9209: { Complement of incomplete Gamma function }
9210: function InvGamma(A, P : Float) : Float; external 'dmath';
9211: { Inverse of incomplete Gamma function }
9212: function Erf(X : Float) : Float; external 'dmath';
9213: { Error function }
9214: function Erfc(X : Float) : Float; external 'dmath';
9215: { Complement of error function }
9216: { ------------------------------------------------------------
9217:   Beta function and related functions
9218:   ------------------------------------------------------------ }
9219: function Beta(X, Y : Float) : Float; external 'dmath';
9220: { Beta function }
9221: function IBeta(A, B, X : Float) : Float; external 'dmath';
9222: { Incomplete Beta function }
9223: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9224: { Inverse of incomplete Beta function }
9225: { ------------------------------------------------------------
9226:   Lambert's function
9227:   ------------------------------------------------------------ }
9228: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9229:  ------------------------------------------------------------
9230:   Binomial distribution
9231:   ------------------------------------------------------------ }
9232: function Binomial(N, K : Integer) : Float; external 'dmath';
9233: { Binomial coefficient C(N,K) }
9234: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9235: { Probability of binomial distribution }
9236: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9237: { Cumulative probability for binomial distrib. }
9238: { ------------------------------------------------------------
9239:   Poisson distribution
9240:   ------------------------------------------------------------ }
9241: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9242: { Probability of Poisson distribution }
9243: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9244: { Cumulative probability for Poisson distrib. }
9245: { ------------------------------------------------------------
9246:   Exponential distribution
9247:   ------------------------------------------------------------ }
9248: function DExpo(A, X : Float) : Float; external 'dmath';
9249: { Density of exponential distribution with parameter A }
9250: function FExpo(A, X : Float) : Float; external 'dmath';
9251: { Cumulative probability function for exponential dist. with parameter A }
9252: { ------------------------------------------------------------
9253:   Standard normal distribution
9254:   ------------------------------------------------------------ }
9255: function DNorm(X : Float) : Float; external 'dmath';
```

```
9256: { Density of standard normal distribution }
9257: function FNorm(X : Float) : Float; external 'dmath';
9258: { Cumulative probability for standard normal distrib. }
9259: function PNorm(X : Float) : Float; external 'dmath';
9260: { Prob(|U| > X) for standard normal distrib. }
9261: function InvNorm(P : Float) : Float; external 'dmath';
9262: { Inverse of standard normal distribution }
9263: { ---------------------------------------------------------------
9264:   Student's distribution
9265:   --------------------------------------------------------------- }
9266: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9267: { Density of Student distribution with Nu d.o.f. }
9268: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9269: { Cumulative probability for Student distrib. with Nu d.o.f. }
9270: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9271: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9272: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9273: { Inverse of Student's t-distribution function }
9274: { ---------------------------------------------------------------
9275:   Khi-2 distribution
9276:   --------------------------------------------------------------- }
9277: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9278: { Density of Khi-2 distribution with Nu d.o.f. }
9279: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9280: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9281: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9282: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9283: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9284: { Inverse of Khi-2 distribution function }
9285: { ---------------------------------------------------------------
9286:   Fisher-Snedecor distribution
9287:   --------------------------------------------------------------- }
9288: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9289: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9290: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9291: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9292: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9293: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9294: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9295: { Inverse of Snedecor's F-distribution function }
9296: { ---------------------------------------------------------------
9297:   Beta distribution
9298:   --------------------------------------------------------------- }
9299: function DBeta(A, B, X : Float) : Float; external 'dmath';
9300: { Density of Beta distribution with parameters A and B }
9301: function FBeta(A, B, X : Float) : Float; external 'dmath';
9302: { Cumulative probability for Beta distrib. with param. A and B }
9303: { ---------------------------------------------------------------
9304:   Gamma distribution
9305:   --------------------------------------------------------------- }
9306: function DGamma(A, B, X : Float) : Float; external 'dmath';
9307: { Density of Gamma distribution with parameters A and B }
9308: function FGamma(A, B, X : Float) : Float; external 'dmath';
9309: { Cumulative probability for Gamma distrib. with param. A and B }
9310: { ---------------------------------------------------------------
9311:   Expression evaluation
9312:   --------------------------------------------------------------- }
9313: function InitEval : Integer; external 'dmath';
9314: { Initializes built-in functions and returns their number }
9315: function Eval(ExpressionString : String) : Float; external 'dmath';
9316: { Evaluates an expression at run-time }
9317: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9318: { Assigns a value to a variable }
9319: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9320: { Adds a function to the parser }
9321: { ---------------------------------------------------------------
9322:   Matrices and linear equations
9323:   --------------------------------------------------------------- }
9324: procedure GaussJordan(A           : TMatrix;
9325:                       Lb, Ub1, Ub2 : Integer;
9326:                       var Det      : Float); external 'dmath';
9327: { Transforms a matrix according to the Gauss-Jordan method }
9328: procedure LinEq(A       : TMatrix;
9329:                 B       : TVector;
9330:                 Lb, Ub  : Integer;
9331:                 var Det : Float); external 'dmath';
9332: { Solves a linear system according to the Gauss-Jordan method }
9333: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9334: { Cholesky factorization of a positive definite symmetric matrix }
9335: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9336: { LU decomposition }
9337: procedure LU_Solve(A      : TMatrix;
9338:                    B      : TVector;
9339:                    Lb, Ub : Integer;
9340:                    X      : TVector); external 'dmath';
9341: { Solution of linear system from LU decomposition }
9342: procedure QR_Decomp(A           : TMatrix;
9343:                     Lb, Ub1, Ub2 : Integer;
9344:                     R           : TMatrix); external 'dmath';
```

```
9345: { QR decomposition }
9346: procedure QR_Solve(Q, R         : TMatrix;
9347:                    B            : TVector;
9348:                    Lb, Ub1, Ub2 : Integer;
9349:                    X            : TVector); external 'dmath';
9350: { Solution of linear system from QR decomposition }
9351: procedure SV_Decomp(A             : TMatrix;
9352:                     Lb, Ub1, Ub2 : Integer;
9353:                     S            : TVector;
9354:                     V            : TMatrix); external 'dmath';
9355: { Singular value decomposition }
9356: procedure SV_SetZero(S      : TVector;
9357:                      Lb, Ub : Integer;
9358:                      Tol    : Float); external 'dmath';
9359: { Set lowest singular values to zero }
9360: procedure SV_Solve(U            : TMatrix;
9361:                    S            : TVector;
9362:                    V            : TMatrix;
9363:                    B            : TVector;
9364:                    Lb, Ub1, Ub2 : Integer;
9365:                    X            : TVector); external 'dmath';
9366: { Solution of linear system from SVD }
9367: procedure SV_Approx(U             : TMatrix;
9368:                     S             : TVector;
9369:                     V             : TMatrix;
9370:                     Lb, Ub1, Ub2 : Integer;
9371:                     A             : TMatrix); external 'dmath';
9372: { Matrix approximation from SVD }
9373: procedure EigenVals(A      : TMatrix;
9374:                     Lb, Ub : Integer;
9375:                     Lambda : TCompVector); external 'dmath';
9376: { Eigenvalues of a general square matrix }
9377: procedure EigenVect(A      : TMatrix;
9378:                     Lb, Ub : Integer;
9379:                     Lambda : TCompVector;
9380:                     V      : TMatrix); external 'dmath';
9381: { Eigenvalues and eigenvectors of a general square matrix }
9382: procedure EigenSym(A      : TMatrix;
9383:                    Lb, Ub : Integer;
9384:                    Lambda : TVector;
9385:                    V      : TMatrix); external 'dmath';
9386: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9387: procedure Jacobi(A               : TMatrix;
9388:                  Lb, Ub, MaxIter : Integer;
9389:                  Tol             : Float;
9390:                  Lambda          : TVector;
9391:                  V               : TMatrix); external 'dmath';
9392: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9393: { ---------------------------------------------------------------
9394:   Optimization
9395:   --------------------------------------------------------------- }
9396: procedure MinBrack(Func                    : TFunc;
9397:                    var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9398: { Brackets a minimum of a function }
9399: procedure GoldSearch(Func           : TFunc;
9400:                      A, B           : Float;
9401:                      MaxIter        : Integer;
9402:                      Tol            : Float;
9403:                      var Xmin, Ymin : Float); external 'dmath';
9404: { Minimization of a function of one variable (golden search) }
9405: procedure LinMin(Func      : TFuncNVar;
9406:                  X, DeltaX : TVector;
9407:                  Lb, Ub    : Integer;
9408:                  var R     : Float;
9409:                  MaxIter   : Integer;
9410:                  Tol       : Float;
9411:                  var F_min : Float); external 'dmath';
9412: { Minimization of a function of several variables along a line }
9413: procedure Newton(Func      : TFuncNVar;
9414:                  HessGrad  : THessGrad;
9415:                  X         : TVector;
9416:                  Lb, Ub    : Integer;
9417:                  MaxIter   : Integer;
9418:                  Tol       : Float;
9419:                  var F_min : Float;
9420:                  G         : TVector;
9421:                  H_inv     : TMatrix;
9422:                  var Det   : Float); external 'dmath';
9423: { Minimization of a function of several variables (Newton's method) }
9424: procedure SaveNewton(FileName : string); external 'dmath';
9425: { Save Newton iterations in a file }
9426: procedure Marquardt(Func      : TFuncNVar;
9427:                     HessGrad  : THessGrad;
9428:                     X         : TVector;
9429:                     Lb, Ub    : Integer;
9430:                     MaxIter   : Integer;
9431:                     Tol       : Float;
9432:                     var F_min : Float;
9433:                     G         : TVector;
```

```
9434:                    H_inv     : TMatrix;
9435:                    var Det   : Float); external 'dmath';
9436: { Minimization of a function of several variables (Marquardt's method) }
9437: procedure SaveMarquardt(FileName : string); external 'dmath';
9438: { Save Marquardt iterations in a file }
9439: procedure BFGS(Func      : TFuncNVar;
9440:                Gradient  : TGradient;
9441:                X         : TVector;
9442:                Lb, Ub    : Integer;
9443:                MaxIter   : Integer;
9444:                Tol       : Float;
9445:                var F_min : Float;
9446:                G         : TVector;
9447:                H_inv     : TMatrix); external 'dmath';
9448: { Minimization of a function of several variables (BFGS method) }
9449: procedure SaveBFGS(FileName : string); external 'dmath';
9450: { Save BFGS iterations in a file }
9451: procedure Simplex(Func      : TFuncNVar;
9452:                   X         : TVector;
9453:                   Lb, Ub    : Integer;
9454:                   MaxIter   : Integer;
9455:                   Tol       : Float;
9456:                   var F_min : Float); external 'dmath';
9457: { Minimization of a function of several variables (Simplex) }
9458: procedure SaveSimplex(FileName : string); external 'dmath';
9459: { Save Simplex iterations in a file }
9460: { ------------------------------------------------------------------
9461:   Nonlinear equations
9462:   ------------------------------------------------------------------ }
9463: procedure RootBrack(Func              : TFunc;
9464:                     var X, Y, FX, FY : Float); external 'dmath';
9465: { Brackets a root of function Func between X and Y }
9466: procedure Bisect(Func     : TFunc;
9467:                  var X, Y : Float;
9468:                  MaxIter  : Integer;
9469:                  Tol      : Float;
9470:                  var F    : Float); external 'dmath';
9471: { Bisection method }
9472: procedure Secant(Func     : TFunc;
9473:                  var X, Y : Float;
9474:                  MaxIter  : Integer;
9475:                  Tol      : Float;
9476:                  var F    : Float); external 'dmath';
9477: { Secant method }
9478: procedure NewtEq(Func, Deriv : TFunc;
9479:                  var X       : Float;
9480:                  MaxIter     : Integer;
9481:                  Tol         : Float;
9482:                  var F       : Float); external 'dmath';
9483: { Newton-Raphson method for a single nonlinear equation }
9484: procedure NewtEqs(Equations : TEquations;
9485:                   Jacobian  : TJacobian;
9486:                   X, F      : TVector;
9487:                   Lb, Ub    : Integer;
9488:                   MaxIter   : Integer;
9489:                   Tol       : Float); external 'dmath';
9490: { Newton-Raphson method for a system of nonlinear equations }
9491: procedure Broyden(Equations : TEquations;
9492:                   X, F      : TVector;
9493:                   Lb, Ub    : Integer;
9494:                   MaxIter   : Integer;
9495:                   Tol       : Float); external 'dmath';
9496: { Broyden's method for a system of nonlinear equations }
9497: { ------------------------------------------------------------------
9498:   Polynomials and rational fractions
9499:   ------------------------------------------------------------------ }
9500: function Poly(X    : Float;
9501:               Coef : TVector;
9502:               Deg  : Integer) : Float; external 'dmath';
9503: { Evaluates a polynomial }
9504: function RFrac(X          : Float;
9505:                Coef       : TVector;
9506:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9507: { Evaluates a rational fraction }
9508: function RootPol1(A, B  : Float;
9509:                   var X : Float) : Integer; external 'dmath';
9510: { Solves the linear equation A + B * X = 0 }
9511: function RootPol2(Coef : TVector;
9512:                   Z    : TCompVector) : Integer; external 'dmath';
9513: { Solves a quadratic equation }
9514: function RootPol3(Coef : TVector;
9515:                   Z    : TCompVector) : Integer; external 'dmath';
9516: { Solves a cubic equation }
9517: function RootPol4(Coef : TVector;
9518:                   Z    : TCompVector) : Integer; external 'dmath';
9519: { Solves a quartic equation }
9520: function RootPol(Coef : TVector;
9521:                  Deg  : Integer;
9522:                  Z    : TCompVector) : Integer; external 'dmath';
```

```
9523: { Solves a polynomial equation }
9524: function SetRealRoots(Deg : Integer;
9525:                       Z   : TCompVector;
9526:                       Tol : Float) : Integer; external 'dmath';
9527: { Set the imaginary part of a root to zero }
9528: procedure SortRoots(Deg : Integer;
9529:                     Z   : TCompVector); external 'dmath';
9530: { Sorts the roots of a polynomial }
9531: { ---------------------------------------------------------------
9532:   Numerical integration and differential equations
9533:   --------------------------------------------------------------- }
9534: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9535: { Integration by trapezoidal rule }
9536: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9537: { Integral from A to B }
9538: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9539: { Integral from 0 to B }
9540: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9541: { Convolution product at time T }
9542: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9543: { Convolution by trapezoidal rule }
9544: procedure RKF45(F                    : TDiffEqs;
9545:                 Neqn                 : Integer;
9546:                 Y, Yp                : TVector;
9547:                 var T                : Float;
9548:                 Tout, RelErr, AbsErr : Float;
9549:                 var Flag             : Integer); external 'dmath';
9550: { Integration of a system of differential equations }
9551: { ---------------------------------------------------------------
9552:   Fast Fourier Transform
9553:   --------------------------------------------------------------- }
9554: procedure FFT(NumSamples        : Integer;
9555:               InArray, OutArray : TCompVector); external 'dmath';
9556: { Fast Fourier Transform }
9557: procedure IFFT(NumSamples        : Integer;
9558:                InArray, OutArray : TCompVector); external 'dmath';
9559: { Inverse Fast Fourier Transform }
9560: procedure FFT_Integer(NumSamples     : Integer;
9561:                       RealIn, ImagIn : TIntVector;
9562:                       OutArray       : TCompVector); external 'dmath';
9563: { Fast Fourier Transform for integer data }
9564: procedure FFT_Integer_Cleanup; external 'dmath';
9565: { Clear memory after a call to FFT_Integer }
9566: procedure CalcFrequency(NumSamples,
9567:                         FrequencyIndex : Integer;
9568:                         InArray        : TCompVector;
9569:                         var FFT        : Complex); external 'dmath';
9570: { Direct computation of Fourier transform }
9571: { ---------------------------------------------------------------
9572:   Random numbers
9573:   --------------------------------------------------------------- }
9574: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9575: { Select generator }
9576: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9577: { Initialize generator }
9578: function IRanGen : RNG_IntType; external 'dmath';
9579: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9580: function IRanGen31 : RNG_IntType; external 'dmath';
9581: { 31-bit random integer in [0 .. 2^31 - 1] }
9582: function RanGen1 : Float; external 'dmath';
9583: { 32-bit random real in [0,1] }
9584: function RanGen2 : Float; external 'dmath';
9585: { 32-bit random real in [0,1) }
9586: function RanGen3 : Float; external 'dmath';
9587: { 32-bit random real in (0,1) }
9588: function RanGen53 : Float; external 'dmath';
9589: { 53-bit random real in [0,1) }
9590: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9591: { Initializes the 'Multiply with carry' random number generator }
9592: function IRanMWC : RNG_IntType; external 'dmath';
9593: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9594: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9595: { Initializes Mersenne Twister generator with a seed }
9596: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9597:                         KeyLength : Word); external 'dmath';
9598: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9599: function IRanMT : RNG_IntType; external 'dmath';
9600: { Random integer from MT generator }
9601: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9602: { Initializes the UVAG generator with a string }
9603: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9604: { Initializes the UVAG generator with an integer }
9605: function IRanUVAG : RNG_IntType; external 'dmath';
9606: { Random integer from UVAG generator }
9607: function RanGaussStd : Float; external 'dmath';
9608: { Random number from standard normal distribution }
9609: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9610: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9611: procedure RanMult(M      : TVector; L      : TMatrix;
```

```
9612:                   Lb, Ub : Integer;
9613:                   X      : TVector); external 'dmath';
9614: { Random vector from multinormal distribution (correlated) }
9615: procedure RanMultIndep(M, S   : TVector;
9616:                        Lb, Ub : Integer;
9617:                        X      : TVector); external 'dmath';
9618: { Random vector from multinormal distribution (uncorrelated) }
9619: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9620: { Initializes Metropolis-Hastings parameters }
9621: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9622: { Returns Metropolis-Hastings parameters }
9623: procedure Hastings(Func      : TFuncNVar;
9624:                    T         : Float;
9625:                    X         : TVector;
9626:                    V         : TMatrix;
9627:                    Lb, Ub    : Integer;
9628:                    Xmat      : TMatrix;
9629:                    X_min     : TVector;
9630:                    var F_min : Float); external 'dmath';
9631: { Simulation of a probability density function by Metropolis-Hastings }
9632: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9633: { Initializes Simulated Annealing parameters }
9634: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9635: { Initializes log file }
9636: procedure SimAnn(Func          : TFuncNVar;
9637:                  X, Xmin, Xmax : TVector;
9638:                  Lb, Ub        : Integer;
9639:                  var F_min     : Float); external 'dmath';
9640: { Minimization of a function of several var. by simulated annealing }
9641: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9642: { Initializes Genetic Algorithm parameters }
9643: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9644: { Initializes log file }
9645: procedure GenAlg(Func          : TFuncNVar;
9646:                  X, Xmin, Xmax : TVector;
9647:                  Lb, Ub        : Integer;
9648:                  var F_min     : Float); external 'dmath';
9649: { Minimization of a function of several var. by genetic algorithm }
9650: { ------------------------------------------------------------------
9651:   Statistics
9652:   ------------------------------------------------------------------ }
9653: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9654: { Mean of sample X }
9655: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9656: { Minimum of sample X }
9657: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9658: { Maximum of sample X }
9659: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9660: { Median of sample X }
9661: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9662: { Standard deviation estimated from sample X }
9663: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9664: { Standard deviation of population }
9665: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9666: { Correlation coefficient }
9667: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9668: { Skewness of sample X }
9669: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9670: { Kurtosis of sample X }
9671: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9672: { Quick sort (ascending order) }
9673: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9674: { Quick sort (descending order) }
9675: procedure Interval(X1, X2             : Float;
9676:                    MinDiv, MaxDiv     : Integer;
9677:                    var Min, Max, Step : Float); external 'dmath';
9678: { Determines an interval for a set of values }
9679: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9680:                     var XMin, XMax, XStep : Float); external 'dmath';
9681: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9682: procedure StudIndep(N1, N2         : Integer;
9683:                     M1, M2, S1, S2 : Float;
9684:                     var T          : Float;
9685:                     var DoF        : Integer); external 'dmath';
9686: { Student t-test for independent samples }
9687: procedure StudPaired(X, Y    : TVector;
9688:                      Lb, Ub  : Integer;
9689:                      var T   : Float;
9690:                      var DoF : Integer); external 'dmath';
9691: { Student t-test for paired samples }
9692: procedure AnOVa1(Ns               : Integer;
9693:                  N                : TIntVector;
9694:                  M, S             : TVector;
9695:                  var V_f, V_r, F  : Float;
9696:                  var DoF_f, DoF_r : Integer); external 'dmath';
9697: { One-way analysis of variance }
9698: procedure AnOVa2(NA, NB, Nobs : Integer;
9699:                  M, S         : TMatrix;
9700:                  V, F         : TVector;
```

```
9701:                  DoF          : TIntVector); external 'dmath';
9702: { Two-way analysis of variance }
9703: procedure Snedecor(N1, N2         : Integer;
9704:                    S1, S2         : Float;
9705:                    var F          : Float;
9706:                    var DoF1, DoF2 : Integer); external 'dmath';
9707: { Snedecor's F-test (comparison of two variances) }
9708: procedure Bartlett(Ns       : Integer;
9709:                    N        : TIntVector;
9710:                    S        : TVector;
9711:                    var Khi2 : Float;
9712:                    var DoF  : Integer); external 'dmath';
9713: { Bartlett's test (comparison of several variances) }
9714: procedure Mann_Whitney(N1, N2     : Integer;
9715:                        X1, X2     : TVector;
9716:                        var U, Eps : Float); external 'dmath';
9717: { Mann-Whitney test}
9718: procedure Wilcoxon(X, Y       : TVector;
9719:                    Lb, Ub     : Integer;
9720:                    var Ndiff  : Integer;
9721:                    var T, Eps : Float); external 'dmath';
9722: { Wilcoxon test }
9723: procedure Kruskal_Wallis(Ns      : Integer;
9724:                          N       : TIntVector;
9725:                          X       : TMatrix;
9726:                          var H   : Float;
9727:                          var DoF : Integer); external 'dmath';
9728: { Kruskal-Wallis test }
9729: procedure Khi2_Conform(N_cls    : Integer;
9730:                        N_estim  : Integer;
9731:                        Obs      : TIntVector;
9732:                        Calc     : TVector;
9733:                        var Khi2 : Float;
9734:                        var DoF  : Integer); external 'dmath';
9735: { Khi-2 test for conformity }
9736: procedure Khi2_Indep(N_lin    : Integer;
9737:                      N_col    : Integer;
9738:                      Obs      : TIntMatrix;
9739:                      var Khi2 : Float;
9740:                      var DoF  : Integer); external 'dmath';
9741: { Khi-2 test for independence }
9742: procedure Woolf_Conform(N_cls   : Integer;
9743:                         N_estim : Integer;
9744:                         Obs     : TIntVector;
9745:                         Calc    : TVector;
9746:                         var G   : Float;
9747:                         var DoF : Integer); external 'dmath';
9748: { Woolf's test for conformity }
9749: procedure Woolf_Indep(N_lin   : Integer;
9750:                       N_col   : Integer;
9751:                       Obs     : TIntMatrix;
9752:                       var G   : Float;
9753:                       var DoF : Integer); external 'dmath';
9754: { Woolf's test for independence }
9755: procedure DimStatClassVector(var C : TStatClassVector;
9756:                              Ub    : Integer); external 'dmath';
9757: { Allocates an array of statistical classes: C[0..Ub] }
9758: procedure Distrib(X       : TVector;
9759:                   Lb, Ub  : Integer;
9760:                   A, B, H : Float;
9761:                   C       : TStatClassVector); external 'dmath';
9762: { Distributes an array X[Lb..Ub] into statistical classes }
9763: { ----------------------------------------------------------------
9764:   Linear / polynomial regression
9765:   ---------------------------------------------------------------- }
9766: procedure LinFit(X, Y   : TVector;
9767:                  Lb, Ub : Integer;
9768:                  B      : TVector;
9769:                  V      : TMatrix); external 'dmath';
9770: { Linear regression : Y = B(0) + B(1) * X }
9771: procedure WLinFit(X, Y, S : TVector;
9772:                   Lb, Ub  : Integer;
9773:                   B       : TVector;
9774:                   V       : TMatrix); external 'dmath';
9775: { Weighted linear regression : Y = B(0) + B(1) * X }
9776: procedure SVDLinFit(X, Y   : TVector;
9777:                     Lb, Ub : Integer;
9778:                     SVDTol : Float;
9779:                     B      : TVector;
9780:                     V      : TMatrix); external 'dmath';
9781: { Unweighted linear regression by singular value decomposition }
9782: procedure WSVDLinFit(X, Y, S : TVector;
9783:                      Lb, Ub  : Integer;
9784:                      SVDTol  : Float;
9785:                      B       : TVector;
9786:                      V       : TMatrix); external 'dmath';
9787: { Weighted linear regression by singular value decomposition }
9788: procedure MulFit(X            : TMatrix;
9789:                  Y            : TVector;
```

```
9790:                 Lb, Ub, Nvar : Integer;
9791:                 ConsTerm     : Boolean;
9792:                 B            : TVector;
9793:                 V            : TMatrix); external 'dmath';
9794: { Multiple linear regression by Gauss-Jordan method }
9795: procedure WMulFit(X            : TMatrix;
9796:                  Y, S         : TVector;
9797:                  Lb, Ub, Nvar : Integer;
9798:                  ConsTerm     : Boolean;
9799:                  B            : TVector;
9800:                  V            : TMatrix); external 'dmath';
9801: { Weighted multiple linear regression by Gauss-Jordan method }
9802: procedure SVDFit(X            : TMatrix;
9803:                 Y            : TVector;
9804:                 Lb, Ub, Nvar : Integer;
9805:                 ConsTerm     : Boolean;
9806:                 SVDTol       : Float;
9807:                 B            : TVector;
9808:                 V            : TMatrix); external 'dmath';
9809: { Multiple linear regression by singular value decomposition }
9810: procedure WSVDFit(X            : TMatrix;
9811:                  Y, S         : TVector;
9812:                  Lb, Ub, Nvar : Integer;
9813:                  ConsTerm     : Boolean;
9814:                  SVDTol       : Float;
9815:                  B            : TVector;
9816:                  V            : TMatrix); external 'dmath';
9817: { Weighted multiple linear regression by singular value decomposition }
9818: procedure PolFit(X, Y        : TVector;
9819:                 Lb, Ub, Deg : Integer;
9820:                 B           : TVector;
9821:                 V           : TMatrix); external 'dmath';
9822: { Polynomial regression by Gauss-Jordan method }
9823: procedure WPolFit(X, Y, S     : TVector;
9824:                  Lb, Ub, Deg : Integer;
9825:                  B           : TVector;
9826:                  V           : TMatrix); external 'dmath';
9827: { Weighted polynomial regression by Gauss-Jordan method }
9828: procedure SVDPolFit(X, Y        : TVector;
9829:                    Lb, Ub, Deg : Integer;
9830:                    SVDTol      : Float;
9831:                    B           : TVector;
9832:                    V           : TMatrix); external 'dmath';
9833: { Unweighted polynomial regression by singular value decomposition }
9834: procedure WSVDPolFit(X, Y, S     : TVector;
9835:                     Lb, Ub, Deg : Integer;
9836:                     SVDTol      : Float;
9837:                     B           : TVector;
9838:                     V           : TMatrix); external 'dmath';
9839: { Weighted polynomial regression by singular value decomposition }
9840: procedure RegTest(Y, Ycalc : TVector;
9841:                  LbY, UbY : Integer;
9842:                  V        : TMatrix;
9843:                  LbV, UbV : Integer;
9844:                  var Test : TRegTest); external 'dmath';
9845: { Test of unweighted regression }
9846: procedure WRegTest(Y, Ycalc, S : TVector;
9847:                   LbY, UbY    : Integer;
9848:                   V           : TMatrix;
9849:                   LbV, UbV    : Integer;
9850:                   var Test    : TRegTest); external 'dmath';
9851: { Test of weighted regression }
9852: { ------------------------------------------------------------------
9853:   Nonlinear regression
9854:   ------------------------------------------------------------------ }
9855: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9856: { Sets the optimization algorithm for nonlinear regression }
9857: function GetOptAlgo : TOptAlgo; external 'dmath';
9858: { Returns the optimization algorithm }
9859: procedure SetMaxParam(N : Byte); external 'dmath';
9860: { Sets the maximum number of regression parameters for nonlinear regression }
9861: function GetMaxParam : Byte; external 'dmath';
9862: { Returns the maximum number of regression parameters for nonlinear regression }
9863: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9864: { Sets the bounds on the I-th regression parameter }
9865: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9866: { Returns the bounds on the I-th regression parameter }
9867: procedure NLFit(RegFunc   : TRegFunc;
9868:                DerivProc : TDerivProc;
9869:                X, Y      : TVector;
9870:                Lb, Ub    : Integer;
9871:                MaxIter   : Integer;
9872:                Tol       : Float;
9873:                B         : TVector;
9874:                FirstPar,
9875:                LastPar   : Integer;
9876:                V         : TMatrix); external 'dmath';
9877: { Unweighted nonlinear regression }
9878: procedure WNLFit(RegFunc   : TRegFunc;
```

```
9879:                 DerivProc : TDerivProc;
9880:                 X, Y, S   : TVector;
9881:                 Lb, Ub    : Integer;
9882:                 MaxIter   : Integer;
9883:                 Tol       : Float;
9884:                 B         : TVector;
9885:                 FirstPar,
9886:                 LastPar   : Integer;
9887:                 V         : TMatrix); external 'dmath';
9888: { Weighted nonlinear regression }
9889: procedure SetMCFile(FileName : String); external 'dmath';
9890: { Set file for saving MCMC simulations }
9891: procedure SimFit(RegFunc   : TRegFunc;
9892:                 X, Y      : TVector;
9893:                 Lb, Ub    : Integer;
9894:                 B         : TVector;
9895:                 FirstPar,
9896:                 LastPar   : Integer;
9897:                 V         : TMatrix); external 'dmath';
9898: { Simulation of unweighted nonlinear regression by MCMC }
9899: procedure WSimFit(RegFunc   : TRegFunc;
9900:                  X, Y, S   : TVector;
9901:                  Lb, Ub    : Integer;
9902:                  B         : TVector;
9903:                  FirstPar,
9904:                  LastPar   : Integer;
9905:                  V         : TMatrix); external 'dmath';
9906: { Simulation of weighted nonlinear regression by MCMC }
9907: { ---------------------------------------------------------------
9908:   Nonlinear regression models
9909:   --------------------------------------------------------------- }
9910: procedure FracFit(X, Y       : TVector;
9911:                  Lb, Ub     : Integer;
9912:                  Deg1, Deg2 : Integer;
9913:                  ConsTerm   : Boolean;
9914:                  MaxIter    : Integer;
9915:                  Tol        : Float;
9916:                  B          : TVector;
9917:                  V          : TMatrix); external 'dmath';
9918: { Unweighted fit of rational fraction }
9919: procedure WFracFit(X, Y, S    : TVector;
9920:                   Lb, Ub     : Integer;
9921:                   Deg1, Deg2 : Integer;
9922:                   ConsTerm   : Boolean;
9923:                   MaxIter    : Integer;
9924:                   Tol        : Float;
9925:                   B          : TVector;
9926:                   V          : TMatrix); external 'dmath';
9927: { Weighted fit of rational fraction }
9928:
9929: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9930: { Returns the value of the rational fraction at point X }
9931: procedure ExpFit(X, Y         : TVector;
9932:                 Lb, Ub, Nexp : Integer;
9933:                 ConsTerm     : Boolean;
9934:                 MaxIter      : Integer;
9935:                 Tol          : Float;
9936:                 B            : TVector;
9937:                 V            : TMatrix); external 'dmath';
9938: { Unweighted fit of sum of exponentials }
9939: procedure WExpFit(X, Y, S      : TVector;
9940:                  Lb, Ub, Nexp : Integer;
9941:                  ConsTerm     : Boolean;
9942:                  MaxIter      : Integer;
9943:                  Tol          : Float;
9944:                  B            : TVector;
9945:                  V            : TMatrix); external 'dmath';
9946: { Weighted fit of sum of exponentials }
9947: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9948: { Returns the value of the regression function at point X }
9949: procedure IncExpFit(X, Y     : TVector;
9950:                    Lb, Ub    : Integer;
9951:                    ConsTerm  : Boolean;
9952:                    MaxIter   : Integer;
9953:                    Tol       : Float;
9954:                    B         : TVector;
9955:                    V         : TMatrix); external 'dmath';
9956: { Unweighted fit of model of increasing exponential }
9957: procedure WIncExpFit(X, Y, S  : TVector;
9958:                     Lb, Ub   : Integer;
9959:                     ConsTerm : Boolean;
9960:                     MaxIter  : Integer;
9961:                     Tol      : Float;
9962:                     B        : TVector;
9963:                     V        : TMatrix); external 'dmath';
9964: { Weighted fit of increasing exponential }
9965: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9966: { Returns the value of the regression function at point X }
9967: procedure ExpLinFit(X, Y    : TVector;
```

```
 9968:                    Lb, Ub  : Integer;
 9969:                    MaxIter : Integer;
 9970:                    Tol     : Float;
 9971:                    B       : TVector;
 9972:                    V       : TMatrix); external 'dmath';
 9973: { Unweighted fit of the "exponential + linear" model }
 9974: procedure WExpLinFit(X, Y, S : TVector;
 9975:                     Lb, Ub  : Integer;
 9976:                     MaxIter : Integer;
 9977:                     Tol     : Float;
 9978:                     B       : TVector;
 9979:                     V       : TMatrix); external 'dmath';
 9980: { Weighted fit of the "exponential + linear" model }
 9981:
 9982: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9983: { Returns the value of the regression function at point X }
 9984: procedure MichFit(X, Y    : TVector;
 9985:                   Lb, Ub  : Integer;
 9986:                   MaxIter : Integer;
 9987:                   Tol     : Float;
 9988:                   B       : TVector;
 9989:                   V       : TMatrix); external 'dmath';
 9990: { Unweighted fit of Michaelis equation }
 9991: procedure WMichFit(X, Y, S : TVector;
 9992:                    Lb, Ub  : Integer;
 9993:                    MaxIter : Integer;
 9994:                    Tol     : Float;
 9995:                    B       : TVector;
 9996:                    V       : TMatrix); external 'dmath';
 9997: { Weighted fit of Michaelis equation }
 9998: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9999: { Returns the value of the Michaelis equation at point X }
10000: procedure MintFit(X, Y    : TVector;
10001:                   Lb, Ub  : Integer;
10002:                   MintVar : TMintVar;
10003:                   Fit_S0  : Boolean;
10004:                   MaxIter : Integer;
10005:                   Tol     : Float;
10006:                   B       : TVector;
10007:                   V       : TMatrix); external 'dmath';
10008: { Unweighted fit of the integrated Michaelis equation }
10009: procedure WMintFit(X, Y, S : TVector;
10010:                    Lb, Ub  : Integer;
10011:                    MintVar : TMintVar;
10012:                    Fit_S0  : Boolean;
10013:                    MaxIter : Integer;
10014:                    Tol     : Float;
10015:                    B       : TVector;
10016:                    V       : TMatrix); external 'dmath';
10017: { Weighted fit of the integrated Michaelis equation }
10018: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10019: { Returns the value of the integrated Michaelis equation at point X }
10020: procedure HillFit(X, Y    : TVector;
10021:                   Lb, Ub  : Integer;
10022:                   MaxIter : Integer;
10023:                   Tol     : Float;
10024:                   B       : TVector;
10025:                   V       : TMatrix); external 'dmath';
10026: { Unweighted fit of Hill equation }
10027: procedure WHillFit(X, Y, S : TVector;
10028:                    Lb, Ub  : Integer;
10029:                    MaxIter : Integer;
10030:                    Tol     : Float;
10031:                    B       : TVector;
10032:                    V       : TMatrix); external 'dmath';
10033: { Weighted fit of Hill equation }
10034: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10035: { Returns the value of the Hill equation at point X }
10036: procedure LogiFit(X, Y     : TVector;
10037:                   Lb, Ub   : Integer;
10038:                   ConsTerm : Boolean;
10039:                   General  : Boolean;
10040:                   MaxIter  : Integer;
10041:                   Tol      : Float;
10042:                   B        : TVector;
10043:                   V        : TMatrix); external 'dmath';
10044: { Unweighted fit of logistic function }
10045: procedure WLogiFit(X, Y, S  : TVector;
10046:                    Lb, Ub   : Integer;
10047:                    ConsTerm : Boolean;
10048:                    General  : Boolean;
10049:                    MaxIter  : Integer;
10050:                    Tol      : Float;
10051:                    B        : TVector;
10052:                    V        : TMatrix); external 'dmath';
10053: { Weighted fit of logistic function }
10054: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10055: { Returns the value of the logistic function at point X }
10056: procedure PKFit(X, Y    : TVector;
```

```
10057:                Lb, Ub  : Integer;
10058:                MaxIter : Integer;
10059:                Tol     : Float;
10060:                B       : TVector;
10061:                V       : TMatrix); external 'dmath';
10062: { Unweighted fit of the acid-base titration curve }
10063: procedure WPKFit(X, Y, S : TVector;
10064:                  Lb, Ub  : Integer;
10065:                  MaxIter : Integer;
10066:                  Tol     : Float;
10067:                  B       : TVector;
10068:                  V       : TMatrix); external 'dmath';
10069: { Weighted fit of the acid-base titration curve }
10070: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10071: { Returns the value of the acid-base titration function at point X }
10072: procedure PowFit(X, Y    : TVector;
10073:                  Lb, Ub  : Integer;
10074:                  MaxIter : Integer;
10075:                  Tol     : Float;
10076:                  B       : TVector;
10077:                  V       : TMatrix); external 'dmath';
10078: { Unweighted fit of power function }
10079: procedure WPowFit(X, Y, S : TVector;
10080:                   Lb, Ub  : Integer;
10081:                   MaxIter : Integer;
10082:                   Tol     : Float;
10083:                   B       : TVector;
10084:                   V       : TMatrix); external 'dmath';
10085: { Weighted fit of power function }
10086:
10087: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10088: { Returns the value of the power function at point X }
10089: procedure GammaFit(X, Y    : TVector;
10090:                    Lb, Ub  : Integer;
10091:                    MaxIter : Integer;
10092:                    Tol     : Float;
10093:                    B       : TVector;
10094:                    V       : TMatrix); external 'dmath';
10095: { Unweighted fit of gamma distribution function }
10096: procedure WGammaFit(X, Y, S : TVector;
10097:                     Lb, Ub  : Integer;
10098:                     MaxIter : Integer;
10099:                     Tol     : Float;
10100:                     B       : TVector;
10101:                     V       : TMatrix); external 'dmath';
10102: { Weighted fit of gamma distribution function }
10103: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10104: { Returns the value of the gamma distribution function at point X }
10105: { ---------------------------------------------------------------
10106:   Principal component analysis
10107:   --------------------------------------------------------------- }
10108: procedure VecMean(X            : TMatrix;
10109:                   Lb, Ub, Nvar : Integer;
10110:                   M            : TVector); external 'dmath';
10111: { Computes the mean vector M from matrix X }
10112: procedure VecSD(X            : TMatrix;
10113:                 Lb, Ub, Nvar : Integer;
10114:                 M, S         : TVector); external 'dmath';
10115: { Computes the vector of standard deviations S from matrix X }
10116: procedure MatVarCov(X            : TMatrix;
10117:                     Lb, Ub, Nvar : Integer;
10118:                     M            : TVector;
10119:                     V            : TMatrix); external 'dmath';
10120: { Computes the variance-covariance matrix V from matrix X }
10121: procedure MatCorrel(V    : TMatrix;
10122:                     Nvar : Integer;
10123:                     R    : TMatrix); external 'dmath';
10124: { Computes the correlation matrix R from the var-cov matrix V }
10125: procedure PCA(R      : TMatrix;
10126:               Nvar   : Integer;
10127:               Lambda : TVector;
10128:               C, Rc  : TMatrix); external 'dmath';
10129: { Performs a principal component analysis of the correlation matrix R }
10130: procedure ScaleVar(X            : TMatrix;
10131:                    Lb, Ub, Nvar : Integer;
10132:                    M, S         : TVector;
10133:                    Z            : TMatrix); external 'dmath';
10134: { Scales a set of variables by subtracting means and dividing by SD's }
10135: procedure PrinFac(Z            : TMatrix;
10136:                   Lb, Ub, Nvar : Integer;
10137:                   C, F         : TMatrix); external 'dmath';
10138: { Computes principal factors }
10139: { ---------------------------------------------------------------
10140:   Strings
10141:   --------------------------------------------------------------- }
10142: function LTrim(S : String) : String; external 'dmath';
10143: { Removes leading blanks }
10144: function RTrim(S : String) : String; external 'dmath';
10145: { Removes trailing blanks }
```

```
10146: function Trim(S : String) : String; external 'dmath';
10147: { Removes leading and trailing blanks }
10148: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10149: { Returns a string made of character C repeated N times }
10150: function RFill(S : String; L : Byte) : String; external 'dmath';
10151: { Completes string S with trailing blanks for a total length L }
10152: function LFill(S : String; L : Byte) : String; external 'dmath';
10153: { Completes string S with leading blanks for a total length L }
10154: function CFill(S : String; L : Byte) : String; external 'dmath';
10155: { Centers string S on a total length L }
10156: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10157: { Replaces in string S all the occurences of C1 by C2 }
10158: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10159: { Extracts a field from a string }
10160: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10161: { Parses a string into its constitutive fields }
10162: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10163: { Sets the numeric format }
10164: function FloatStr(X : Float) : String; external 'dmath';
10165: { Converts a real to a string according to the numeric format }
10166: function IntStr(N : LongInt) : String; external 'dmath';
10167: { Converts an integer to a string }
10168: function CompStr(Z : Complex) : String; external 'dmath';
10169: { Converts a complex number to a string }
10170: {$IFDEF DELPHI}
10171: function StrDec(S : String) : String; external 'dmath';
10172: { Set decimal separator to the symbol defined in SysUtils }
10173: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10174: { Test if a string represents a number and returns it in X }
10175: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10176: { Reads a floating point number from an Edit control }
10177: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10178: { Writes a floating point number in a text file }
10179: {$ENDIF}
10180: { ------------------------------------------------------------------
10181:   BGI / Delphi graphics
10182:   ------------------------------------------------------------------ }
10183: function InitGraphics
10184: {$IFDEF DELPHI}
10185: (Width, Height : Integer) : Boolean;
10186: {$ELSE}
10187: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10188: { Enters graphic mode }
10189: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10190:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10191: { Sets the graphic window }
10192: procedure SetOxScale(Scale                : TScale;
10193:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10194: { Sets the scale on the Ox axis }
10195: procedure SetOyScale(Scale                : TScale;
10196:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10197: { Sets the scale on the Oy axis }
10198: procedure GetOxScale(var Scale                : TScale;
10199:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10200: { Returns the scale on the Ox axis }
10201: procedure GetOyScale(var Scale                : TScale;
10202:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10203: { Returns the scale on the Oy axis }
10204: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10205: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10206: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10207: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10208: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10209: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10210: {$IFNDEF DELPHI}
10211: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10212: { Sets the font for the main graph title }
10213: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10214: { Sets the font for the Ox axis (title and labels) }
10215: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10216: { Sets the font for the Oy axis (title and labels) }
10217: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10218: { Sets the font for the legends }
10219: procedure SetClipping(Clip : Boolean); external 'dmath';
10220: { Determines whether drawings are clipped at the current viewport
10221:   boundaries, according to the value of the Boolean parameter Clip }
10222: {$ENDIF}
10223: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10224: { Plots the horizontal axis }
10225: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10226: { Plots the vertical axis }
10227: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10228: { Plots a grid on the graph }
10229: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10230: { Writes the title of the graph }
10231: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10232: { Sets the maximum number of curves and re-initializes their parameters }
10233: procedure SetPointParam
10234: {$IFDEF DELPHI}
```

```
10235: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10236: {$ELSE}
10237: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10238: { Sets the point parameters for curve # CurvIndex }
10239: procedure SetLineParam
10240: {$IFDEF DELPHI}
10241: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10242: {$ELSE}
10243: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10244: { Sets the line parameters for curve # CurvIndex }
10245: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10246: { Sets the legend for curve # CurvIndex }
10247: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10248: { Sets the step for curve # CurvIndex }
10249: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10250: procedure GetPointParam
10251: {$IFDEF DELPHI}
10252: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10253: {$ELSE}
10254: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10255: { Returns the point parameters for curve # CurvIndex }
10256: procedure GetLineParam
10257: {$IFDEF DELPHI}
10258: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10259: {$ELSE}
10260: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10261: { Returns the line parameters for curve # CurvIndex }
10262: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10263: { Returns the legend for curve # CurvIndex }
10264: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10265: { Returns the step for curve # CurvIndex }
10266: {$IFDEF DELPHI}
10267: procedure PlotPoint(Canvas    : TCanvas;
10268:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
10269: {$ELSE}
10270: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10271: {$ENDIF}
10272: { Plots a point on the screen }
10273: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10274:                     X, Y                  : TVector;
10275:                     Lb, Ub, CurvIndex     : Integer); external 'dmath';
10276: { Plots a curve }
10277: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10278:                                  X, Y, S              : TVector;
10279:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10280: { Plots a curve with error bars }
10281: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10282:                    Func                  : TFunc;
10283:                    Xmin, Xmax            : Float;
10284:                    {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10285:                    CurvIndex             : Integer); external 'dmath';
10286: { Plots a function }
10287: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10288:                       NCurv              : Integer;
10289:                       ShowPoints, ShowLines : Boolean); external 'dmath';
10290: { Writes the legends for the plotted curves }
10291: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10292:                  Nx, Ny, Nc            : Integer;
10293:                  X, Y, Z               : TVector;
10294:                  F                     : TMatrix); external 'dmath';
10295: { Contour plot }
10296: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10297: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10298: function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
10299: function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
10300: {$IFNDEF DELPHI}
10301: procedure LeaveGraphics; external 'dmath';
10302: { Quits graphic mode }
10303: {$ENDIF}
10304: { ---------------------------------------------------------------
10305:   LaTeX graphics
10306:   --------------------------------------------------------------- }
10307: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
10308:                           Header            : Boolean) : Boolean; external 'dmath';
10309: { Initializes the LaTeX file }
10310: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10311: { Sets the graphic window }
10312: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10313: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10314: { Sets the scale on the Ox axis }
10315: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10316: { Sets the scale on the Oy axis }
10317: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10318: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10319: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10320: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10321: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10322: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10323: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
```

```
10324: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10325: { Sets the maximum number of curves and re-initializes their parameters }
10326: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10327: { Sets the point parameters for curve # CurvIndex }
10328: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10329:                           Width : Float; Smooth : Boolean); external 'dmath';
10330: { Sets the line parameters for curve # CurvIndex }
10331: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10332: { Sets the legend for curve # CurvIndex }
10333: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10334: { Sets the step for curve # CurvIndex }
10335: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10336: { Plots a curve }
10337: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10338:                                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10339: { Plots a curve with error bars }
10340: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10341:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
10342: { Plots a function }
10343: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10344: { Writes the legends for the plotted curves }
10345: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
10346: { Contour plot }
10347: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10348: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10349:
10350: //****************************************************unit uPSI_SynPdf;
10351:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10352:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10353:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10354:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10355:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10356:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10357:  //Function _GetCharCount( Text : PAnsiChar) : integer
10358:  //Procedure L2R( W : PWideChar; L : integer)
10359:  Function PdfCoord( MM : single) : integer
10360:  Function CurrentPrinterPaperSize : TPDFPaperSize
10361:  Function CurrentPrinterRes : TPoint
10362:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10363:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10364:  Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10365:  Const('Usp10','String 'usp10.dll
10366:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10367:    'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10368:   AddTypeS('TScriptState_set', 'set of TScriptState_enum
10369: //*********************************************************************
10370:
10371: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10372: begin
10373:  Procedure PMrandomize( I : word)
10374:  Function PMrandom : longint
10375:  Function Rrand : extended
10376:  Function Irand( N : word) : word
10377:  Function Brand( P : extended) : boolean
10378:  Function Nrand : extended
10379: end;
10380:
10381: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10382: begin
10383:   Function Endian( x : LongWord) : LongWord
10384:   Function Endian64( x : Int64) : Int64
10385:   Function spRol( x : LongWord; y : Byte) : LongWord
10386:   Function spRor( x : LongWord; y : Byte) : LongWord
10387:   Function Ror64( x : Int64; y : Byte) : Int64
10388: end;
10389:
10390: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10391: begin
10392:  Procedure ClearModules
10393:  Procedure ReadMapFile( Fname : string)
10394:  Function AddressInfo( Address : dword) : string
10395: end;
10396:
10397: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10398: begin
10399:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10400:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10401:    +'teByOther, tpExecuteByOther )
10402:   TTarPermissions', 'set of TTarPermission
10403:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10404:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10405:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10406:   TTarModes', 'set of TTarMode
10407:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10408:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10409:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10410:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10411:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10412:   SIRegister_TTarArchive(CL);
```

```
10413:   SIRegister_TTarWriter(CL);
10414:  Function PermissionString( Permissions : TTarPermissions) : STRING
10415:  Function ConvertFilename( Filename : STRING) : STRING
10416:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10417:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10418:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10419: end;
10420:
10421:
10422: //************************************************unit uPSI_TlHelp32;
10423: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10424: begin
10425:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10426:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10427:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10428:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10429:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10430:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10431:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10432:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10433:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10434:   AddTypeS('THeapList32', 'tagHEAPLIST32
10435:  Const('HF32_DEFAULT','LongInt'( 1);
10436:  Const('HF32_SHARED','LongInt'( 2);
10437:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10438:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10439:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10440:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10441:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10442:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10443:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10444:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10445:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10446:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10447:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10448:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10449:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10450:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10451:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10452:    +'aPri : Longint; dwFlags : DWORD; end
10453:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10454:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10455:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10456:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10457: end;
10458:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10459:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10460:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10461:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10462:  Const('EWX_LOGOFF','LongInt'( 0);
10463:  Const('EWX_SHUTDOWN','LongInt'( 1);
10464:  Const('EWX_REBOOT','LongInt'( 2);
10465:  Const('EWX_FORCE','LongInt'( 4);
10466:  Const('EWX_POWEROFF','LongInt'( 8);
10467:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10468:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10469:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10470:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10471:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10472:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10473:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10474:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10475:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10476:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10477:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10478:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10479:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10480:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10481:  Function GetDesktopWindow : HWND
10482:  Function GetParent( hWnd : HWND) : HWND
10483:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10484:  Function GetTopWindow( hWnd : HWND) : HWND
10485:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10486:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10487:  //Delphi DFM
10488:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10489:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10490:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10491:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10492:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10493:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10494:  Function OpenIcon( hWnd : HWND) : BOOL
10495:  Function CloseWindow( hWnd : HWND) : BOOL
10496:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10497:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10498:  Function IsWindowVisible( hWnd : HWND) : BOOL
10499:  Function IsIconic( hWnd : HWND) : BOOL
10500:  Function AnyPopup : BOOL
10501:  Function BringWindowToTop( hWnd : HWND) : BOOL
```

```
10502:  Function IsZoomed( hWnd : HWND) : BOOL
10503:  Function IsWindow( hWnd : HWND) : BOOL
10504:  Function IsMenu( hMenu : HMENU) : BOOL
10505:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10506:  Function DestroyWindow( hWnd : HWND) : BOOL
10507:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10508:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10509:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10510:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10511:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10512:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10513:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10514:
10515: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10516: begin
10517:  const('ShowSetupDialogOptLong','String '--setup
10518:  PrimaryConfPathOptLong','String '--primary-config-path=
10519:  PrimaryConfPathOptShort','String '--pcp=
10520:  SecondaryConfPathOptLong','String '--secondary-config-path=
10521:  SecondaryConfPathOptShort','String '--scp=
10522:  NoSplashScreenOptLong','String '--no-splash-screen
10523:  NoSplashScreenOptShort','String '--nsc
10524:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10525:  SkipLastProjectOpt','String '--skip-last-project
10526:  DebugLogOpt','String '--debug-log=
10527:  DebugLogOptEnable','String '--debug-enable=
10528:  LanguageOpt','String '--language=
10529:  LazarusDirOpt','String '--lazarusdir=
10530:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10531:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10532:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10533:  Function IsHelpRequested : Boolean
10534:  Function IsVersionRequested : boolean
10535:  Function GetLanguageSpecified : string
10536:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10537:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10538:  Procedure ParseNoGuiCmdLineParams
10539:  Function ExtractCmdLineFilenames : TStrings
10540: end;
10541:
10542:
10543: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10544: begin
10545:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10546:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10547:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10548:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10549:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10550:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10551:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10552:  Function DirPathExists( DirectoryName : string) : boolean
10553:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10554:  Function ExtractFileNameOnly( const AFilename : string) : string
10555:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10556:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10557:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10558:  Function ForceDirectory( DirectoryName : string) : boolean
10559:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10560:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10561:  Function FileIsText( const AFilename : string) : boolean
10562:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10563:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10564:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10565:  Function TrimFilename( const AFilename : string) : string
10566:  Function ResolveDots( const AFilename : string) : string
10567:  Procedure ForcePathDelims( var FileName : string)
10568:  Function GetForcedPathDelims( const FileName : string) : String
10569:  Function CleanAndExpandFilename( const Filename : string) : string
10570:  Function CleanAndExpandDirectory( const Filename : string) : string
10571:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10572:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10573:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10574:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
10575:  Function FileIsInPath( const Filename, Path : string) : boolean
10576:  Function AppendPathDelim( const Path : string) : string
10577:  Function ChompPathDelim( const Path : string) : string
10578:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10579:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10580:  Function MinimizeSearchPath( const SearchPath : string) : string
10581:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10582:  (*Function FileExistsUTF8( const Filename : string) : boolean
10583:  Function FileAgeUTF8( const FileName : string) : Longint
10584:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10585:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10586:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10587:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10588:  Procedure FindCloseUTF8( var F : TSearchrec)
```

```
10589:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10590:  Function FileGetAttrUTF8( const FileName : String) : Longint
10591:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10592:  Function DeleteFileUTF8( const FileName : String) : Boolean
10593:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10594:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10595:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10596:  Function GetCurrentDirUTF8 : String
10597:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10598:  Function CreateDirUTF8( const NewDir : String) : Boolean
10599:  Function RemoveDirUTF8( const Dir : String) : Boolean
10600:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10601:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10602:  Function FileCreateUTF8( const FileName : string) : THandle;
10603:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10604:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10605:  Function FileSizeUtf8( const Filename : string) : int64
10606:  Function GetFileDescription( const AFilename : string) : string
10607:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10608:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10609:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10610:  Function IsUNCPath( const Path : String) : Boolean
10611:  Function ExtractUNCVolume( const Path : String) : String
10612:  Function ExtractFileRoot( FileName : String) : String
10613:  Function GetDarwinSystemFilename( Filename : string) : string
10614:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10615:  Function StrToCmdLineParam( const Param : string) : string
10616:  Function MergeCmdLineParams( ParamList : TStrings) : string
10617:  Procedure InvalidateFileStateCache( const Filename : string)
10618:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10619:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10620:  Function FindAllDocs(const Root, extmask: string): TStringlist;
10621:  Function ReadFileToString( const Filename : string) : string
10622:  procedure Inc1(var X: longint; N: Longint);
10623:
10624:  type
10625:   TCopyFileFlag = ( cffOverwriteFile,
10626:     cffCreateDestDirectory, cffPreserveTime );
10627:   TCopyFileFlags = set of TCopyFileFlag;*)
10628:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10629:   TCopyFileFlags', 'set of TCopyFileFlag
10630:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10631: end;
10632:
10633: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10634: begin
10635:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10636:   SIRegister_TMask(CL);
10637:   SIRegister_TParseStringList(CL);
10638:   SIRegister_TMaskList(CL);
10639:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10640:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10641:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10642:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10643: end;
10644:
10645: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10646: begin
10647:   //PShellHookInfo', '^TShellHookInfo // will not work
10648:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10649:   SHELLHOOKINFO', 'TShellHookInfo
10650:   LPSHELLHOOKINFO', 'PShellHookInfo
10651:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10652:   SIRegister_TJvShellHook(CL);
10653:  Function InitJvShellHooks : Boolean
10654:  Procedure UnInitJvShellHooks
10655: end;
10656:
10657: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10658: begin
10659:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10660:    +', dcHasSetSel, dcWantTab, dcNative )
10661:   TDlgCodes', 'set of TDlgCode
10662:  'dcWantMessage',' dcWantAllKeys);
10663:   SIRegister_IJvExControl(CL);
10664:   SIRegister_IJvDenySubClassing(CL);
10665:   SIRegister_TStructPtrMessage(CL);
10666:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10667:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10668:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10669:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10670:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10671:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10672:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10673:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10674:  Function GetFocusedControl( AControl : TControl) : TWinControl
10675:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10676:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10677:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
```

```
10678:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10679:   SIRegister_TJvExControl(CL);
10680:   SIRegister_TJvExWinControl(CL);
10681:   SIRegister_TJvExCustomControl(CL);
10682:   SIRegister_TJvExGraphicControl(CL);
10683:   SIRegister_TJvExHintWindow(CL);
10684:   SIRegister_TJvExPubGraphicControl(CL);
10685: end;
10686:
10687: (*----------------------------------------------------------------------------*)
10688: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10689: begin
10690:  Procedure EncodeStream( Input, Output : TStream)
10691:  Procedure DecodeStream( Input, Output : TStream)
10692:  Function EncodeString1( const Input : string) : string
10693:  Function DecodeString1( const Input : string) : string
10694: end;
10695:
10696: (*----------------------------------------------------------------------------*)
10697: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10698: begin
10699:   SIRegister_TWebAppRegInfo(CL);
10700:   SIRegister_TWebAppRegList(CL);
10701:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10702:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10703:  Procedure UnregisterWebApp( const AProgID : string)
10704:  Function FindRegisteredWebApp( const AProgID : string) : string
10705:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10706:  'sUDPPort','String 'UDPPort
10707: end;
10708:
10709: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10710: begin
10711:  // TStringDynArray', 'array of string
10712:  Function GetEnvVarValue( const VarName : string) : string
10713:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10714:  Function DeleteEnvVar( const VarName : string) : Integer
10715:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
       BufSize:Int):Int;
10716:  Function ExpandEnvVars( const Str : string) : string
10717:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10718:  Procedure GetAllEnvVarNames( const Names : TStrings);
10719:  Function GetAllEnvVarNames1 : TStringDynArray;
10720:  Function EnvBlockSize : Integer
10721:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10722:   SIRegister_TPJEnvVarsEnumerator(CL);
10723:   SIRegister_TPJEnvVars(CL);
10724:   FindClass('TOBJECT'),'EPJEnvVars
10725:   FindClass('TOBJECT'),'EPJEnvVars
10726:  //Procedure Register
10727: end;
10728:
10729: (*----------------------------------------------------------------------------*)
10730: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10731: begin
10732:  'cOneSecInMS','LongInt'( 1000);
10733:  //'cDefTimeSlice','LongInt'( 50);
10734:  //'cDefMaxExecTime',' cOneMinInMS);
10735:  'cAppErrorMask','LongInt'( 1 shl 29);
10736:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10737:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10738:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10739:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10740:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10741:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10742:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10743:   SIRegister_TPJCustomConsoleApp(CL);
10744:   SIRegister_TPJConsoleApp(CL);
10745: end;
10746:
10747: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10748: begin
10749:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10750:   t_encoding', '( uuencode, base64, mime )
10751:  Function internet_date( date : TDateTime) : string
10752:  Function lookup_hostname( const hostname : string) : longint
10753:  Function my_hostname : string
10754:  Function my_ip_address : longint
10755:  Function ip2string( ip_address : longint) : string
10756:  Function resolve_hostname( ip : longint) : string
10757:  Function address_from( const s : string; count : integer) : string
10758:  Function encode_base64( data : TStream) : TStringList
10759:  Function decode_base64( source : TStringList) : TMemoryStream
10760:  Function posn( const s, t : string; count : integer) : integer
10761:  Function poscn( c : char; const s : string; n : integer) : integer
10762:  Function filename_of( const s : string) : string
10763:  //Function trim( const s : string) : string
10764:  //Procedure setlength( var s : string; l : byte)
10765:  Function TimeZoneBias : longint
```

```
10766:  Function eight2seven_quoteprint( const s : string) : string
10767:  Function eight2seven_german( const s : string) : string
10768:  Function seven2eight_quoteprint( const s : string) : string end;
10769:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10770:  Function socketerror : cint
10771:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10772:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10773:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10774:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10775:  Function fplisten( s : cint; backlog : cint) : cint
10776:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10777:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10778:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10779:  Function NetAddrToStr( Entry : in_addr) : String
10780:  Function HostAddrToStr( Entry : in_addr) : String
10781:  Function StrToHostAddr( IP : String) : in_addr
10782:  Function StrToNetAddr( IP : String) : in_addr
10783:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10784:   cint8', 'shortint
10785:   cuint8', 'byte
10786:   cchar', 'cint8
10787:   cschar', 'cint8
10788:   cuchar', 'cuint8
10789:   cint16', 'smallint
10790:   cuint16', 'word
10791:   cshort', 'cint16
10792:   csshort', 'cint16
10793:   cushort', 'cuint16
10794:   cint32', 'longint
10795:   cuint32', 'longword
10796:   cint', 'cint32
10797:   csint', 'cint32
10798:   cuint', 'cuint32
10799:   csigned', 'cint
10800:   cunsigned', 'cuint
10801:   cint64', 'int64
10802:   clonglong', 'cint64
10803:   cslonglong', 'cint64
10804:   cbool', 'longbool
10805:   cfloat', 'single
10806:   cdouble', 'double
10807:   clongdouble', 'extended
10808:
10809: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10810: begin
10811:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10812:   SIRegister_TlkJSONdotnetclass(CL);
10813:   SIRegister_TlkJSONbase(CL);
10814:   SIRegister_TlkJSONnumber(CL);
10815:   SIRegister_TlkJSONstring(CL);
10816:   SIRegister_TlkJSONboolean(CL);
10817:   SIRegister_TlkJSONnull(CL);
10818:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10819:    +'se; data : TObject; var Continue : Boolean)
10820:   SIRegister_TlkJSONcustomlist(CL);
10821:   SIRegister_TlkJSONlist(CL);
10822:   SIRegister_TlkJSONobjectmethod(CL);
10823:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10824:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10825:   SIRegister_TlkHashTable(CL);
10826:   SIRegister_TlkBalTree(CL);
10827:   SIRegister_TlkJSONobject(CL);
10828:   SIRegister_TlkJSON(CL);
10829:   SIRegister_TlkJSONstreamed(CL);
10830:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10831: end;
10832:
10833: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10834: begin
10835:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10836:   SIRegister_TZSortedList(CL);
10837:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10838:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10839:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10840:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10841:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10842:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10843:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10844:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10845:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10846:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10847:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10848:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10849:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10850:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10851:  Function StrToBoolEx( Str : string) : Boolean
10852:  Function BoolToStrEx( Bool : Boolean) : String
10853:  Function IsIpAddr( const Str : string) : Boolean
10854:  Function zSplitString( const Str, Delimiters : string) : TStrings
```

```
10855:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10856:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10857:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10858:  Function FloatToSQLStr( Value : Extended) : string
10859:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10860:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10861:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10862:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10863:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10864:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10865:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10866:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10867:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10868:  Function BytesToVar( const Value : TByteDynArray) : Variant
10869:  Function VarToBytes( const Value : Variant) : TByteDynArray
10870:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10871:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10872:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10873:  Function EncodeCString( const Value : string) : string
10874:  Function DecodeCString( const Value : string) : string
10875:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10876:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10877:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
10878:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
       SubVersion:Integer):Int;
10879:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10880:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10881:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10882:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10883:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10884:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10885:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10886:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10887:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10888:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10889: end;
10890:
10891: unit uPSI_ZEncoding;
10892:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10893:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10894:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10895:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10896:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10897:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10898:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10899:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10900:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10901:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10902:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10903:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10904:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10905:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10906:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10907:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10908:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10909:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10910:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10911:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10912:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10913:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10914:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10915:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10916:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10917:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10918:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10919:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10920:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10921:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10922:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10923:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10924:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10925:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10926:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10927:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10928:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10929:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10930:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10931:  Function ZDefaultSystemCodePage : Word
10932:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10933:
10934:
10935: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10936: begin
10937:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10938:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10939:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10940:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10941:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
```

```
10942:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10943:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10944:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10945:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10946:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10947:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
10948:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10949:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10950:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10951:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10952:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10953:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10954:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10955:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10956:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10957:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10958:  ('ilIdentity','2 RPC_C_IMP_LEVEL_IDENTIFY);
10959:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10960:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10961:  ('EOAC_NONE','LongWord').SetUInt( $0);
10962:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10963:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10964:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10965:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10966:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10967:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10968:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10969:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10970:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10971:   FindClass('TOBJECT'),'EBoldCom
10972:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10973:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10974:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10975:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10976:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10977:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10978:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10979:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10980:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10981:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10982:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10983:  Function BoldCreateGUID : TGUID
10984:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
10985:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
10986:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10987:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10988: end;
10989:
10990: (*---------------------------------------------------------------------------*)
10991: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10992: begin
10993:  Function ParseISODate( s : string) : TDateTime
10994:  Function ParseISODateTime( s : string) : TDateTime
10995:  Function ParseISOTime( str : string) : TDateTime
10996: end;
10997:
10998: (*---------------------------------------------------------------------------*)
10999: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
11000: begin
11001:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
11002:  Function BoldCreateGUIDWithBracketsAsString : string
11003: end;
11004:
11005: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
11006: begin
11007:   FindClass('TOBJECT'),'TBoldFileHandler
11008:   FindClass('TOBJECT'),'TBoldDiskFileHandler
11009:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
11010:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
11011:   SIRegister_TBoldFileHandler(CL);
11012:   SIRegister_TBoldDiskFileHandler(CL);
11013:  Procedure BoldCloseAllFilehandlers
11014:  Procedure BoldRemoveUnchangedFilesFromEditor
11015:  Function BoldFileHandlerList : TBoldObjectArray
11016:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
11017: end;
11018:
11019: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
11020: begin
11021:   PCharArr', 'array of PChar
11022:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
11023:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
11024:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
11025:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
11026:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
```

```
11027:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
11028:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
11029:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
11030:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
11031:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
11032:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
       Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
11033:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
11034: end;
11035:
11036: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
11037: begin
11038:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
11039:   SIRegister_TfrmBoldQueryUser(CL);
11040:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
11041: end;
11042:
11043: (*--------------------------------------------------------------------------*)
11044: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
11045: begin
11046:  //('befIsInDisplayList',' BoldElementFlag0);
11047:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
11048:  //('befFollowerSelected',' BoldElementFlag2);
11049:   FindClass('TOBJECT'),'TBoldQueue
11050:   FindClass('TOBJECT'),'TBoldQueueable
11051:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11052:   SIRegister_TBoldQueueable(CL);
11053:   SIRegister_TBoldQueue(CL);
11054:  Function BoldQueueFinalized : Boolean
11055:  Function BoldInstalledQueue : TBoldQueue
11056: end;
11057:
11058: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
11059: begin
11060:  const mmPerInch','Extended').setExtended( 25.4);
11061:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11062:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11063:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11064:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11065:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11066:   TBarLineType', '( white, black, black_half )
11067:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11068:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11069:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
11070:   TCheckSumMethod', '( csmNone, csmModulo10 )
11071:   SIRegister_TAsBarcode(CL);
11072:  Function CheckSumModulo10( const data : string) : string
11073:  Function ConvertMmToPixelsX( const Value : Double) : Integer
11074:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11075:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11076:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11077: end;
11078:
11079: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11080: begin
11081:    THomogeneousByteVector', 'array[0..3] of Byte
11082:   THomogeneousWordVector', 'array[0..3] of Word
11083:   THomogeneousIntVector', 'array[0..3] of Integer
11084:   THomogeneousFltVector', 'array[0..3] of single
11085:   THomogeneousDblVector', 'array[0..3] of double
11086:   THomogeneousExtVector', 'array[0..3] of extended
11087:   TAffineByteVector', 'array[0..2] of Byte
11088:   TAffineWordVector', 'array[0..2] of Word
11089:   TAffineIntVector', 'array[0..2] of Integer
11090:   TAffineFltVector', 'array[0..2] of single
11091:   TAffineDblVector', 'array[0..2] of double
11092:   TAffineExtVector', 'array[0..2] of extended
11093:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11094:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11095:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11096:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11097:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11098:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11099:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11100:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11101:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11102:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11103:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11104:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11105:   TMatrix4b', 'THomogeneousByteMatrix
11106:   TMatrix4w', 'THomogeneousWordMatrix
11107:   TMatrix4i', 'THomogeneousIntMatrix
11108:   TMatrix4f', 'THomogeneousFltMatrix
11109:   TMatrix4d', 'THomogeneousDblMatrix
11110:   TMatrix4e', 'THomogeneousExtMatrix
11111:   TMatrix3b', 'TAffineByteMatrix
11112:   TMatrix3w', 'TAffineWordMatrix
```

```
11113:   TMatrix3i', 'TAffineIntMatrix
11114:   TMatrix3f', 'TAffineFltMatrix
11115:   TMatrix3d', 'TAffineDblMatrix
11116:   TMatrix3e', 'TAffineExtMatrix
11117:   //'PMatrix', '^TMatrix // will not work
11118:   TMatrixGL', 'THomogeneousFltMatrix
11119:   THomogeneousMatrix', 'THomogeneousFltMatrix
11120:   TAffineMatrix', 'TAffineFltMatrix
11121:   TQuaternion', 'record Vector : TVector4f; end
11122:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11123:   +'ger; Height : Integer; end
11124:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11125:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11126:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11127:  'EPSILON','Extended').setExtended( 1E-100);
11128:  'EPSILON2','Extended').setExtended( 1E-50);
11129:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11130:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11131:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11132:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11133:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11134:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11135:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11136:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11137:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11138:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11139:  Function VectorLength( V : array of Single) : Single
11140:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11141:  Procedure VectorNegate( V : array of Single)
11142:  Function VectorNorm( V : array of Single) : Single
11143:  Function VectorNormalize( V : array of Single) : Single
11144:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11145:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11146:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11147:  Procedure VectorScale( V : array of Single; Factor : Single)
11148:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11149:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11150:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11151:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11152:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11153:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11154:  Procedure MatrixAdjoint( var M : TMatrixGL)
11155:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11156:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11157:  Function MatrixDeterminant( M : TMatrixGL) : Single
11158:  Procedure MatrixInvert( var M : TMatrixGL)
11159:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11160:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11161:  Procedure MatrixTranspose( var M : TMatrixGL)
11162:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11163:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11164:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11165:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11166:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11167:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11168:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11169:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11170:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11171:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11172:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11173:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11174:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11175:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11176:  Function MakeAffineVector( V : array of Single) : TAffineVector
11177:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11178:  Function MakeVector( V : array of Single) : TVectorGL
11179:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11180:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11181:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11182:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11183:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11184:  Function ArcCosGL( X : Extended) : Extended
11185:  Function ArcSinGL( X : Extended) : Extended
11186:  Function ArcTan2GL( Y, X : Extended) : Extended
11187:  Function CoTanGL( X : Extended) : Extended
11188:  Function DegToRadGL( Degrees : Extended) : Extended
11189:  Function RadToDegGL( Radians : Extended) : Extended
11190:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11191:  Function TanGL( X : Extended) : Extended
11192:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11193:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11194:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11195:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11196:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11197:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11198: end;
11199:
11200:
11201: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
```

```
11202: begin
11203:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11204:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11205:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11206:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11207:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11208:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11209:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11210:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11211:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11212:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11213:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11214:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11215:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11216:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11217:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11218:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11219:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11220:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11221:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11222:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11223:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11224:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11225:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11226:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11227:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11228:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
       SaveTo:TStrings):Bool;
11229:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11230: end;
11231:
11232: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11233: begin
11234:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11235:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11236:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11237:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11238:   FindClass('TOBJECT'),'EInvalidParam
11239:  Function IsDCOMInstalled : Boolean
11240:  Function IsDCOMEnabled : Boolean
11241:  Function GetDCOMVersion : string
11242:  Function GetMDACVersion : string
11243:  Function GetMDACVersion2 : string
11244:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11245:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11246:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11247:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11248:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11249:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11250:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11251:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11252:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11253:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11254:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11255:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11256:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11257:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11258: end;
11259:
11260:
11261: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11262: begin
11263:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11264:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11265:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11266:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11267:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11268:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11269:  DegPerCycle','Extended').setExtended( 360.0);
11270:  DegPerGrad','Extended').setExtended( 0.9);
11271:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11272:  GradPerCycle','Extended').setExtended( 400.0);
11273:  GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
11274:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11275:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11276:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11277:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11278:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11279:  CyclePerGrad','Extended').setExtended( 0.0025);
11280:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11281:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11282:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11283:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11284:  Function MakePercentage( const Step, Max : Longint) : Longint
11285:  Function CelsiusToKelvin( const T : double) : double
```

```
11286:  Function CelsiusToFahrenheit( const T : double) : double
11287:  Function KelvinToCelsius( const T : double) : double
11288:  Function KelvinToFahrenheit( const T : double) : double
11289:  Function FahrenheitToCelsius( const T : double) : double
11290:  Function FahrenheitToKelvin( const T : double) : double
11291:  Function CycleToDeg( const Cycles : double) : double
11292:  Function CycleToGrad( const Cycles : double) : double
11293:  Function CycleToRad( const Cycles : double) : double
11294:  Function DegToCycle( const Degrees : double) : double
11295:  Function DegToGrad( const Degrees : double) : double
11296:  Function DegToRad( const Degrees : double) : double
11297:  Function GradToCycle( const Grads : double) : double
11298:  Function GradToDeg( const Grads : double) : double
11299:  Function GradToRad( const Grads : double) : double
11300:  Function RadToCycle( const Radians : double) : double
11301:  Function RadToDeg( const Radians : double) : double
11302:  Function RadToGrad( const Radians : double) : double
11303:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11304:  Function DmsToRad( const D, M : Integer; const S : double) : double
11305:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11306:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11307:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11308:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11309:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11310:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11311:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11312:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11313:  Function CmToInch( const Cm : double) : double
11314:  Function InchToCm( const Inch : double) : double
11315:  Function FeetToMetre( const Feet : double) : double
11316:  Function MetreToFeet( const Metre : double) : double
11317:  Function YardToMetre( const Yard : double) : double
11318:  Function MetreToYard( const Metre : double) : double
11319:  Function NmToKm( const Nm : double) : double
11320:  Function KmToNm( const Km : double) : double
11321:  Function KmToSm( const Km : double) : double
11322:  Function SmToKm( const Sm : double) : double
11323:  Function LitreToGalUs( const Litre : double) : double
11324:  Function GalUsToLitre( const GalUs : double) : double
11325:  Function GalUsToGalCan( const GalUs : double) : double
11326:  Function GalCanToGalUs( const GalCan : double) : double
11327:  Function GalUsToGalUk( const GalUs : double) : double
11328:  Function GalUkToGalUs( const GalUk : double) : double
11329:  Function LitreToGalCan( const Litre : double) : double
11330:  Function GalCanToLitre( const GalCan : double) : double
11331:  Function LitreToGalUk( const Litre : double) : double
11332:  Function GalUkToLitre( const GalUk : double) : double
11333:  Function KgToLb( const Kg : double) : double
11334:  Function LbToKg( const Lb : double) : double
11335:  Function KgToOz( const Kg : double) : double
11336:  Function OzToKg( const Oz : double) : double
11337:  Function CwtUsToKg( const Cwt : double) : double
11338:  Function CwtUkToKg( const Cwt : double) : double
11339:  Function KaratToKg( const Karat : double) : double
11340:  Function KgToCwtUs( const Kg : double) : double
11341:  Function KgToCwtUk( const Kg : double) : double
11342:  Function KgToKarat( const Kg : double) : double
11343:  Function KgToSton( const Kg : double) : double
11344:  Function KgToLton( const Kg : double) : double
11345:  Function StonToKg( const STon : double) : double
11346:  Function LtonToKg( const Lton : double) : double
11347:  Function QrUsToKg( const Qr : double) : double
11348:  Function QrUkToKg( const Qr : double) : double
11349:  Function KgToQrUs( const Kg : double) : double
11350:  Function KgToQrUk( const Kg : double) : double
11351:  Function PascalToBar( const Pa : double) : double
11352:  Function PascalToAt( const Pa : double) : double
11353:  Function PascalToTorr( const Pa : double) : double
11354:  Function BarToPascal( const Bar : double) : double
11355:  Function AtToPascal( const At : double) : double
11356:  Function TorrToPascal( const Torr : double) : double
11357:  Function KnotToMs( const Knot : double) : double
11358:  Function HpElectricToWatt( const HpE : double) : double
11359:  Function HpMetricToWatt( const HpM : double) : double
11360:  Function MsToKnot( const ms : double) : double
11361:  Function WattToHpElectric( const W : double) : double
11362:  Function WattToHpMetric( const W : double) : double
11363:  function getBigPI: string;    //PI of 1000 numbers
11364:
11365: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11366: begin
11367:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11368:  Procedure CDCopyFile( const FileName, DestName : string)
11369:  Procedure CDMoveFile( const FileName, DestName : string)
11370:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11371:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11372:  Function CDGetTempDir : string
11373:  Function CDGetFileSize( FileName : string) : longint
11374:  Function GetFileTime( FileName : string) : longint
```

```
11375:  Function GetShortName( FileName : string) : string
11376:  Function GetFullName( FileName : string) : string
11377:  Function WinReboot : boolean
11378:  Function WinDir : String
11379:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11380:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11381:  Function devExecutor : TdevExecutor
11382: end;
11383:
11384: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11385: begin
11386:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11387:  Procedure Associate( Index : integer)
11388:  Procedure UnAssociate( Index : integer)
11389:  Function IsAssociated( Index : integer) : boolean
11390:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11391:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11392:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11393:  procedure RefreshIcons;
11394: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11395: function MergColor(Colors: Array of TColor): TColor;
11396: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11397: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11398: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11399: function GetInverseColor(AColor: TColor): TColor;
11400: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11401: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11402: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11403: Procedure GetSystemMenuFont(Font: TFont);
11404: end;
11405:
11406: //**************************unit uPSI_JvHLParser;*******************************
11407: function IsStringConstant(const St: string): Boolean;
11408: function IsIntConstant(const St: string): Boolean;
11409: function IsRealConstant(const St: string): Boolean;
11410: function IsIdentifier(const ID: string): Boolean;
11411: function GetStringValue(const St: string): string;
11412: procedure ParseString(const S: string; Ss: TStrings);
11413: function IsStringConstantW(const St: WideString): Boolean;
11414: function IsIntConstantW(const St: WideString): Boolean;
11415: function IsRealConstantW(const St: WideString): Boolean;
11416: function IsIdentifierW(const ID: WideString): Boolean;
11417: function GetStringValueW(const St: WideString): WideString;
11418: procedure ParseStringW(const S: WideString; Ss: TStrings);
11419:
11420:
11421: //**************************unit uPSI_JclMapi;*******************************
11422:
11423: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11424: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11425: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11426: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11427: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11428:
11429: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11430: begin
11431:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11432:  Function BuildType1Message( ADomain, AHost : String) : String
11433:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11434:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11435:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11436: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11437: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11438: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11439: end;
11440:
11441: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11442: begin
11443: ('IpAny','LongWord').SetUInt( $00000000);
11444:  IpLoopBack','LongWord').SetUInt( $7F000001);
11445:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11446:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11447:  PortAny','LongWord( $0000);
11448:  SocketMaxConnections','LongInt'( 5);
11449:  TIpAddr', 'LongWord
11450:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11451:  Function HostToNetLong( HostLong : LongWord) : LongWord
11452:  Function HostToNetShort( HostShort : Word) : Word
11453:  Function NetToHostLong( NetLong : LongWord) : LongWord
11454:  Function NetToHostShort( NetShort : Word) : Word
11455:  Function StrToIp( Ip : string) : TIpAddr
11456:  Function IpToStr( Ip : TIpAddr) : string
11457: end;
11458:
11459: (*----------------------------------------------------------------------------*)
11460: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
```

```
11461: begin
11462:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11463:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11464:    +'amSha1, AlsmtpClientAuthAutoSelect )
11465:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11466:   SIRegister_TAlSmtpClient(CL);
11467: end;
11468:
11469: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11470: begin
11471: 'TBitNo', 'Integer
11472:   TStByteNo', 'Integer
11473: TStationNo', 'Integer
11474: TInOutNo', 'Integer
11475: TIo', '( EE, AA, NE, NA )
11476: TBitSet', 'set of TBitNo
11477: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11478: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11479: TBitAddr', 'LongInt
11480: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11481: TByteAddr', 'SmallInt
11482: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11483:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11484:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11485:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11486:  Function BitAddrToStr( Value : TBitAddr) : string
11487:  Function StrToBitAddr( const Value : string) : TBitAddr
11488:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11489:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11490:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11491:  Function ByteAddrToStr( Value : TByteAddr) : string
11492:  Function StrToByteAddr( const Value : string) : TByteAddr
11493:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11494:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11495:  Function InOutStateToStr( State : TInOutState) : string
11496:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11497:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11498: end;
11499:
11500: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11501: begin
11502: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11503:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11504: DpmiPmVector', 'Int64
11505:  'DInterval','LongInt'( 1000);
11506:  //'DEnabled','Boolean')BoolToStr( True);
11507:  'DIntFreq','string' if64
11508:  //'DMessages','Boolean if64);
11509:   SIRegister_TwdxCustomTimer(CL);
11510:   SIRegister_TwdxTimer(CL);
11511:   SIRegister_TwdxRtcTimer(CL);
11512:   SIRegister_TCustomIntTimer(CL);
11513:   SIRegister_TIntTimer(CL);
11514:   SIRegister_TRtcIntTimer(CL);
11515:  Function RealNow : TDateTime
11516:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11517:  Function DateTimeToMs( Time : TDateTime) : LongInt
11518: end;
11519:
11520: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11521: begin
11522: TIdSyslogPRI', 'Integer
11523: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11524:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11525:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11526:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11527:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11528: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11529:   SIRegister_TIdSysLogMsgPart(CL);
11530:   SIRegister_TIdSysLogMessage(CL);
11531:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11532:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11533:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11534:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11535:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11536:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11537: end;
11538:
11539: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11540: begin
11541:  'UWhitespace','String '(?:\s*)
11542:  Function StripSpaces( const AText : string) : string
11543:  Function CharCount( const AText : string; Ch : Char) : Integer
11544:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11545:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11546: end;
11547:
11548:
```

```
11549: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11550: begin
11551:  ExtPascalVersion','String '0.9.8
11552:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11553:    +'Opera, brKonqueror, brMobileSafari )
11554:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11555:   AddTypeS('TExtProcedure', 'Procedure
11556:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11557:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11558:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11559:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11560:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11561:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11562:  Function StrToJS( const S : string; UseBR : boolean) : string
11563:  Function CaseOf( const S : string; const Cases : array of string) : integer
11564:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11565:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11566:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11567:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11568:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11569:  Function IsUpperCase( S : string) : boolean
11570:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11571:  Function BeautifyCSS( const AStyle : string) : string
11572:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11573:  Function JSDateToDateTime( JSDate : string) : TDateTime
11574: end;
11575:
11576: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11577: begin
11578:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11579:   TSHDeleteOptions', 'set of TSHDeleteOption
11580:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11581:   TSHRenameOptions', 'set of TSHRenameOption
11582:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11583:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11584:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11585:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11586:   TEnumFolderFlags', 'set of TEnumFolderFlag
11587:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11588:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11589:    +'IEnumIdList; Folder : IShellFolder; end
11590:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11591:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11592:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11593:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11594:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11595:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11596:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11597:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11598:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11599:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11600:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11601:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11602:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11603:  Function SHFreeMem( var P : Pointer) : Boolean
11604:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11605:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11606:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11607:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11608:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11609:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11610:  Function PidlFree( var IdList : PItemIdList) : Boolean
11611:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11612:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11613:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11614:  Function PidlToPath( IdList : PItemIdList) : string
11615:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11616:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11617:   PShellLink', '^TShellLink // will not work
11618:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11619:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11620:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11621:  Procedure ShellLinkFree( var Link : TShellLink)
11622:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11623:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11624:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11625:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11626:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11627:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11628:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11629:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11630:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11631:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11632:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11633:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11634:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11635:  Function ShellOpenAs( const FileName : string) : Boolean
11636:  Function ShellRasDial( const EntryName : string) : Boolean
11637:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
```

```
11638:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11639:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11640:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11641:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11642:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11643:  Function OemKeyScan( wOemChar : Word) : DWORD
11644:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11645: end;
11646:
11647: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11648: begin
11649:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11650:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11651:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11652:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11653:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11654:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11655:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11656:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11657:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11658:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11659:  Function xmlValidName( const Text : UnicodeString) : Boolean
11660:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11661:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11662:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11663:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11664:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11665:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11666:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11667:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11668:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11669:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11670:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11671:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11672:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11673:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11674:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11675:  Procedure SelfTestcXMLFunctions
11676: end;
11677:
11678: (*----------------------------------------------------------------------------*)
11679: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11680: begin
11681:  Function AWaitCursor : IUnknown
11682:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11683:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11684:  Function YesNo( const ACaption, AMsg : string) : boolean
11685:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11686:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11687:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11688:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11689:  Procedure GetSystemPaths( Strings : TStrings)
11690:  Procedure MakeEditNumeric( EditHandle : integer)
11691: end;
11692:
11693: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11694: begin
11695:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11696:  'BI_YUY2','LongWord( $32595559);
11697:  'BI_UYVY','LongWord').SetUInt( $59565955);
11698:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11699:  'BI_YVU9','LongWord').SetUInt( $39555659);
11700:  'BI_YUV12','LongWord( $30323449);
11701:  'BI_Y8','LongWord').SetUInt( $20203859);
11702:  'BI_Y211','LongWord').SetUInt( $31313259);
11703:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11704:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11705: end;
11706:
11707: (*----------------------------------------------------------------------------*)
11708: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11709: begin
11710:  'WM_USER','LongWord').SetUInt( $0400);
11711:  'WM_CAP_START','LongWord').SetUint($0400);
11712:  'WM_CAP_END','longword').SetUint($0400+85);
11713:  //WM_CAP_START+  85
11714:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11715:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11716:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11717:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11718:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11719:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11720:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11721:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11722:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11723:  Function capGetUserData( hwnd : THandle) : LongInt
11724:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11725:  Function capDriverDisconnect( hwnd : THandle) : LongInt
```

```
11726:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11727:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11728:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11729:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11730:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11731:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11732:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11733:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11734:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11735:  Function capEditCopy( hwnd : THandle) : LongInt
11736:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11737:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11738:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11739:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11740:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11741:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11742:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11743:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11744:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11745:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11746:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11747:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11748:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11749:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11750:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11751:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11752:  Function capGrabFrame( hwnd : THandle) : LongInt
11753:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11754:  Function capCaptureSequence( hwnd : THandle) : LongInt
11755:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11756:  Function capCaptureStop( hwnd : THandle) : LongInt
11757:  Function capCaptureAbort( hwnd : THandle) : LongInt
11758:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11759:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11760:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11761:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11762:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11763:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11764:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11765:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11766:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11767:  Function capPalettePaste( hwnd : THandle) : LongInt
11768:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11769:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11770:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11771:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11772:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11773:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11774:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11775:   //PCapStatus', '^TCapStatus // will not work
11776:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11777:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11778:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11779:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11780:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11781:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11782:    +' wNumAudioAllocated : WORD; end
11783:   //PCaptureParms', '^TCaptureParms // will not work
11784:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11785:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11786:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11787:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11788:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11789:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11790:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11791:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11792:    +'he : BOOL; AVStreamMaster : WORD; end
11793:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11794:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11795:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11796:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11797:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
      : Integer; hwndParent : THandle; nID : Integer) : THandle
11798:  Function
      capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11799:  'IDS_CAP_BEGIN','LongInt'( 300);
11800:  'IDS_CAP_END','LongInt'( 301);
11801:  'IDS_CAP_INFO','LongInt'( 401);
11802:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11803:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11804:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11805:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11806:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11807:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11808:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11809:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11810:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11811:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11812:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
```

```
11813:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11814:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11815:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11816:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11817:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11818:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11819:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11820:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11821:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11822:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11823:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11824:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11825:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11826:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11827:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11828:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11829:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11830:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11831:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11832:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11833:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11834:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11835:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11836:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11837:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11838:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11839:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11840:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11841:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11842:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11843:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11844:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11845:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11846:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11847:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11848:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11849:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11850:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11851:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11852:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11853:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11854:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11855:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11856:  'AVICAP32','String 'AVICAP32.dll
11857: end;
11858:
11859: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11860: begin
11861:  Function AlBoolToInt( Value : Boolean) : Integer
11862:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11863:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11864:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11865:  Function ALInc( var x : integer; Count : integer) : Integer
11866:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11867:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11868:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11869:  Function ALIsInteger(const S: AnsiString): Boolean;
11870:  function ALIsDecimal(const S: AnsiString): boolean;
11871:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11872:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11873:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11874:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11875:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11876:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11877:  Function ALRandomStr(const aLength: Longint): AnsiString;
11878:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11879:  Function ALRandomStrU(const aLength: Longint): String;
11880: end;
11881:
11882: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11883: begin
11884:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
       aTrueStr: AnsiString; const aFalseStr : AnsiString)
11885: end;
11886:
11887: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11888: begin
11889:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11890:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11891:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11892:    +'; ullAvailExtendedVirtual : Int64; end
11893:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11894:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11895:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11896:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11897:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11898:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11899:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11900:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
```

```
11901: end;
11902:
11903: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11904: begin
11905:   SIRegister_THandledObject(CL);
11906:   SIRegister_TEvent(CL);
11907:   SIRegister_TMutex(CL);
11908:   SIRegister_TSharedMem(CL);
11909:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11910:  'TRACE_BUFFER','String 'TRACE_BUFFER
11911:  'TRACE_MUTEX','String 'TRACE_MUTEX
11912:   //PTraceEntry', '^TTraceEntry // will not work
11913:   SIRegister_TIPCTracer(CL);
11914:  'MAX_CLIENTS','LongInt'( 6);
11915:  'IPCTIMEOUT','LongInt'( 2000);
11916:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11917:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11918:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11919:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11920:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11921:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11922:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11923:   FindClass('TOBJECT'),'EMonitorActive
11924:   FindClass('TOBJECT'),'TIPCThread
11925:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11926:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11927:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11928:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11929:   TClientFlags', 'set of TClientFlag
11930:   //PEventData', '^TEventData // will not work
11931:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11932:    +'lag; Flags : TClientFlags; end
11933:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11934:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11935:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11936:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11937:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11938:   SIRegister_TIPCEvent(CL);
11939:   //PClientDirRecords', '^TClientDirRecords // will not work
11940:   SIRegister_TClientDirectory(CL);
11941:   TIPCState', '( stInActive, stDisconnected, stConnected )
11942:   SIRegister_TIPCThread(CL);
11943:   SIRegister_TIPCMonitor(CL);
11944:   SIRegister_TIPCClient(CL);
11945:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11946: end;
11947:
11948: (*----------------------------------------------------------------------------*)
11949: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11950: begin
11951:   SIRegister_TAlGSMComm(CL);
11952:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11953:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
      AMessage:AnsiString);
11954:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11955:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
      UseGreekAlphabet:Bool):Widestring;
11956:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11957:  end;
11958:
11959: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11960: begin
11961:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11962:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11963:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11964:   TInternetScheme', 'integer
11965:   TALIPv6Binary', 'array[1..16] of Char;
11966:  // TALIPv6Binary = array[1..16] of ansiChar;
11967:  //   TInternetScheme = Integer;
11968:   SIRegister_TALHTTPRequestHeader(CL);
11969:   SIRegister_TALHTTPCookie(CL);
11970:   SIRegister_TALHTTPCookieCollection(CL);
11971:   SIRegister_TALHTTPResponseHeader(CL);
11972:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11973:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11974: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
      Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11975: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
      Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11976: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
      Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11977:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11978:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11979:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11980:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11981:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11982:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
      ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11983:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
      Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
```

```
11984:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11985:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11986:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11987:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11988:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11989:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11990:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11991:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11992:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11993:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11994:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
11995:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11996:  Function ALZeroIpV6 : TALIPv6Binary
11997:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11998:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
11999:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
12000:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
12001: end;
12002:
12003: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
12004: begin
12005:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
12006:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
12007:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
12008:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
12009:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
12010:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
12011:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
12012:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
12013:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
12014:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
12015:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
12016: end;
12017:
12018: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
12019: begin
12020:   SIRegister_TALEMailHeader(CL);
12021:   SIRegister_TALNewsArticleHeader(CL);
12022:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
12023:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
12024:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
12025:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
12026:  Function AlGenerateInternetMessageID : AnsiString;
12027:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
12028:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
12029:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
12030: end;
12031:
12032: (*----------------------------------------------------------------------------*)
12033: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
12034: begin
12035:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
12036:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
12037:  Function ALgetLocalIPs : TALStrings
12038:  Function ALgetLocalHostName : AnsiString
12039: end;
12040:
12041: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
12042: begin
12043:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
12044:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12045:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
12046:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
12047:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12048:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12049:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
12050:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12051:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
12052: end;
12053:
12054: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
12055: begin
12056:   TStartupInfoA', 'TStartupInfo
12057:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12058:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
```

```
12059:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
12060:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
12061:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
12062:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
12063:  SE_TCB_NAME','String 'SeTcbPrivilege
12064:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
12065:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
12066:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
12067:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
12068:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
12069:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
12070:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
12071:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
12072:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
12073:  SE_BACKUP_NAME','String 'SeBackupPrivilege
12074:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12075:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12076:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12077:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12078:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12079:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12080:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12081:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12082:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12083:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12084:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12085:  Function AlGetEnvironmentString : AnsiString
12086:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12087:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12088:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12089:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12090:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12091: end;
12092:
12093: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12094: begin
12095:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12096:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12097:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12098:  Function ALGetModuleName : ansistring
12099:  Function ALGetModuleFileNameWithoutExtension : ansistring
12100:  Function ALGetModulePath : ansiString
12101:  Function AlGetFileSize( const AFileName : ansistring) : int64
12102:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12103:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12104:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12105:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12106:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12107:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12108:  Function ALFileExists( const Path : ansiString) : boolean
12109:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12110:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12111:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12112:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12113:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12114: end;
12115:
12116: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12117: begin
12118:   NativeInt', 'Integer
12119:   NativeUInt', 'Cardinal
12120:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12121:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12122:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12123:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12124:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12125:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12126:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12127:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12128:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12129:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12130:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12131:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12132:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12133:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12134:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12135:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
```

```
12136:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12137:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12138:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12139:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12140:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12141:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12142:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12143:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12144:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12145:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12146:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12147:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12148:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12149:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12150:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12151:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12152: end;
12153:
12154: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12155: begin
12156:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12157:   FindClass('TOBJECT'),'TALXMLNode
12158:   FindClass('TOBJECT'),'TALXMLNodeList
12159:   FindClass('TOBJECT'),'TALXMLDocument
12160:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12161:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12162:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12163:    +'nst Name : AnsiString; const Attributes : TALStrings)
12164:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12165:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12166:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12167:    +'ntDocType, ntDocFragment, ntNotation )
12168:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12169:   TALXMLDocOptions', 'set of TALXMLDocOption
12170:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12171:   TALXMLParseOptions', 'set of TALXMLParseOption
12172:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12173:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12174:   SIRegister_EALXMLDocError(CL);
12175:   SIRegister_TALXMLNodeList(CL);
12176:   SIRegister_TALXMLNode(CL);
12177:   SIRegister_TALXmlElementNode(CL);
12178:   SIRegister_TALXmlAttributeNode(CL);
12179:   SIRegister_TALXmlTextNode(CL);
12180:   SIRegister_TALXmlDocumentNode(CL);
12181:   SIRegister_TALXmlCommentNode(CL);
12182:   SIRegister_TALXmlProcessingInstrNode(CL);
12183:   SIRegister_TALXmlCDataNode(CL);
12184:   SIRegister_TALXmlEntityRefNode(CL);
12185:   SIRegister_TALXmlEntityNode(CL);
12186:   SIRegister_TALXmlDocTypeNode(CL);
12187:   SIRegister_TALXmlDocFragmentNode(CL);
12188:   SIRegister_TALXmlNotationNode(CL);
12189:   SIRegister_TALXMLDocument(CL);
12190:  cAlXMLUTF8EncodingStr','String 'UTF-8
12191:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12192:  CALNSDelim','String ':
12193:  CALXML','String 'xml
12194:  CALVersion','String 'version
12195:  CALEncoding','String 'encoding
12196:  CALStandalone','String 'standalone
12197:  CALDefaultNodeIndent','String '
12198:  CALXmlDocument','String 'DOCUMENT
12199:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12200:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
       EncodingStr:AnsiString);
12201:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
       ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12202:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
       ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12203:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
       Recurse: Boolean):TalxmlNode
12204:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
       AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12205:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
       AnsiString
12206: end;
12207:
12208: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12209: //based on TEEProc, TeCanvas, TEEngine, TChart
12210: begin
12211:  'TeePiStep','Double').setExtended( Pi / 180.0);
12212:  'TeeDefaultPerspective','LongInt'( 100);
12213:  'TeeMinAngle','LongInt'( 270);
12214:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12215:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
```

```
12216:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12217:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12218:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12219:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12220:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12221:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12222:  'TA_LEFT','LongInt'( 0);
12223:  'TA_RIGHT','LongInt'( 2);
12224:  'TA_CENTER','LongInt'( 6);
12225:  'TA_TOP','LongInt'( 0);
12226:  'TA_BOTTOM','LongInt'( 8);
12227:  'teePATCOPY','LongInt'( 0);
12228:  'NumCirclePoints','LongInt'( 64);
12229:  'teeDEFAULT_CHARSET','LongInt'( 1);
12230:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12231:  'TA_LEFT','LongInt'( 0);
12232:  'bs_Solid','LongInt'( 0);
12233:  'teepf24Bit','LongInt'( 0);
12234:  'teepfDevice','LongInt'( 1);
12235:  'CM_MOUSELEAVE','LongInt'( 10000);
12236:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12237:  'DC_BRUSH','LongInt'( 18);
12238:  'DC_PEN','LongInt'( 19);
12239:   teeCOLORREF', 'LongWord
12240:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12241:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12242:   SIRegister_TFilterRegion(CL);
12243:   SIRegister_IFormCreator(CL);
12244:   SIRegister_TTeeFilter(CL);
12245:   //TFilterClass', 'class of TTeeFilter
12246:   SIRegister_TFilterItems(CL);
12247:   SIRegister_TConvolveFilter(CL);
12248:   SIRegister_TBlurFilter(CL);
12249:   SIRegister_TTeePicture(CL);
12250:   TPenEndStyle', '( esRound, esSquare, esFlat )
12251:   SIRegister_TChartPen(CL);
12252:   SIRegister_TChartHiddenPen(CL);
12253:   SIRegister_TDottedGrayPen(CL);
12254:   SIRegister_TDarkGrayPen(CL);
12255:   SIRegister_TWhitePen(CL);
12256:   SIRegister_TChartBrush(CL);
12257:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12258:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12259:   SIRegister_TView3DOptions(CL);
12260:   FindClass('TOBJECT'),'TTeeCanvas
12261:   TTeeTransparency', 'Integer
12262:   SIRegister_TTeeBlend(CL);
12263:   FindClass('TOBJECT'),'TCanvas3D
12264:   SIRegister_TTeeShadow(CL);
12265:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
        gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12266:   FindClass('TOBJECT'),'TSubGradient
12267:   SIRegister_TCustomTeeGradient(CL);
12268:   SIRegister_TSubGradient(CL);
12269:   SIRegister_TTeeGradient(CL);
12270:   SIRegister_TTeeFontGradient(CL);
12271:   SIRegister_TTeeFont(CL);
12272:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12273:   TCanvasTextAlign', 'Integer
12274:   TTeeCanvasHandle', 'HDC
12275:   SIRegister_TTeeCanvas(CL);
12276:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12277:   SIRegister_TFloatXYZ(CL);
12278:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12279:   TRGB', 'record blue: byte; green: byte; red: byte; end
12280:   {TRGB=packed record
12281:     Blue  : Byte;
12282:     Green : Byte;
12283:     Red   : Byte;
12284:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12285:
12286: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12287:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12288: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12289: TCanvas3DPlane', '( cpX, cpY, cpZ )
12290:   //IInterface', 'IUnknown
12291:   SIRegister_TCanvas3D(CL);
12292:   SIRegister_TTeeCanvas3D(CL);
12293:   TTrianglePoints', 'Array[0..2] of TPoint;
12294:   TFourPoints', 'Array[0..3] of TPoint;
12295:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12296:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12297:  Function Point3D( const x, y, z : Integer) : TPoint3D
12298:  Procedure SwapDouble( var a, b : Double)
12299:  Procedure SwapInteger( var a, b : Integer)
12300:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12301:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12302:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12303:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
```

```
12304:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12305:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12306:  Procedure UnClipCanvas( ACanvas : TCanvas)
12307:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12308:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12309:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12310:  'TeeCharForHeight','String 'W
12311:  'DarkerColorQuantity','Byte').SetUInt( 128);
12312:  'DarkColorQuantity','Byte').SetUInt( 64);
12313:   TButtonGetColorProc', 'Function  : TColor
12314:   SIRegister_TTeeButton(CL);
12315:   SIRegister_TButtonColor(CL);
12316:   SIRegister_TComboFlat(CL);
12317:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12318:  Function TeePoint( const aX, aY : Integer) : TPoint
12319:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12320:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12321:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12322:  Function OrientRectangle( const R : TRect) : TRect
12323:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12324:  Function PolygonBounds( const P : array of TPoint) : TRect
12325:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12326:  Function RGBValue( const Color : TColor) : TRGB
12327:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12328:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12329:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12330:  Function TeeCull( const P : TFourPoints) : Boolean;
12331:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12332:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12333:  Procedure SmoothStretch( Src, Dst : TBitmap);
12334:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12335:  Function TeeDistance( const x, y : Double) : Double
12336:  Function TeeLoadLibrary( const FileName : String) : HInst
12337:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12338:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12339:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12340:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
      Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12341:   SIRegister_ICanvasHyperlinks(CL);
12342:   SIRegister_ICanvasToolTips(CL);
12343:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12344: end;
12345:
12346: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12347: begin
12348:   TOvcHdc', 'Integer
12349:   TOvcHWND', 'Cardinal
12350:   TOvcHdc', 'HDC
12351:   TOvcHWND', 'HWND
12352:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12353:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12354:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12355:  Function DefaultEpoch : Integer
12356:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12357:  Procedure FixRealPrim( P : PChar; DC : Char)
12358:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12359:  Function GetLeftButton : Byte
12360:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12361:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12362:  Function GetShiftFlags : Byte
12363:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12364:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12365:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12366:  Function ovIsForegroundTask : Boolean
12367:  Function ovTrimLeft( const S : string) : string
12368:  Function ovTrimRight( const S : string) : string
12369:  Function ovQuotedStr( const S : string) : string
12370:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12371:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12372:  Function PtrDiff( const P1, P2 : PChar) : Word
12373:  Procedure PtrInc( var P, Delta : Word)
12374:  Procedure PtrDec( var P, Delta : Word)
12375:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12376:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
      SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12377:  Function ovMinI( X, Y : Integer) : Integer
12378:  Function ovMaxI( X, Y : Integer) : Integer
12379:  Function ovMinL( X, Y : LongInt) : LongInt
12380:  Function ovMaxL( X, Y : LongInt) : LongInt
12381:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12382:  Function PartialCompare( const S1, S2 : string) : Boolean
12383:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12384:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12385:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12386:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
      TransparentColor : TColor)
12387:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
      TransparentColor : TColorRef)
12388:  Function ovWidthOf( const R : TRect) : Integer
```

```
12389:  Function ovHeightOf( const R : TRect) : Integer
12390:  Procedure ovDebugOutput( const S : string)
12391:  Function GetArrowWidth( Width, Height : Integer) : Integer
12392:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12393:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12394:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12395:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12396:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12397:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12398:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12399:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12400:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12401:  Function CreateMetaFile( p1 : PChar) : HDC
12402:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12403:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12404:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12405:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12406:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12407:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12408:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12409:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12410:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12411:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12412:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12413:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12414:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
       SrcHeight:Int;Rop:DWORD):BOOL
12415:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12416:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
       SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12417:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12418:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12419:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12420:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12421:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12422:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12423:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12424:  Function UpdateColors( DC : HDC) : BOOL
12425:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12426:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12427:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12428:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12429:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12430:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12431:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
       HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12432:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12433:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12434:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12435:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12436:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12437:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12438:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12439:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12440:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12441:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12442:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12443:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12444:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12445: end;
12446:
12447: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12448: begin
12449:   SIRegister_TOvcAbstractStore(CL);
12450:   //PExPropInfo', '^TExPropInfo // will not work
12451: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12452:   SIRegister_TOvcPropertyList(CL);
12453:   SIRegister_TOvcDataFiler(CL);
12454:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12455:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12456:  Function CreateStoredItem( const CompName, PropName : string) : string
12457:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12458:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12459: end;
12460:
12461: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12462: begin
12463:  'ovsetsize','LongInt'( 16);
12464:  'etSyntax','LongInt'( 0);
12465:  'etSymantic','LongInt'( 1);
12466:  'chCR','Char #13);
12467:  'chLF','Char #10);
12468:  'chLineSeparator',' chCR);
12469:   SIRegister_TCocoError(CL);
12470:   SIRegister_TCommentItem(CL);
12471:   SIRegister_TCommentList(CL);
12472:   SIRegister_TSymbolPosition(CL);
12473: TGenListType', '( glNever, glAlways, glOnError )
```

```
12474: TovBitSet', 'set of Integer
12475:   //PStartTable', '^TStartTable // will not work
12476: 'TovCharSet', 'set of AnsiChar
12477: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12478: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12479: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12480: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12481: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12482:    +'osition; const Data : string; ErrorType : integer)
12483: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12484: TGetCH', 'Function ( pos : longint) : char
12485: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12486:   SIRegister_TCocoRScanner(CL);
12487:   SIRegister_TCocoRGrammar(CL);
12488:  '_EF','Char #0);
12489:  '_TAB','Char').SetString( #09);
12490:  '_CR','Char').SetString( #13);
12491:  '_LF','Char').SetString( #10);
12492:  '_EL','').SetString( _CR);
12493:  '_EOF','Char').SetString( #26);
12494:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12495:  'minErrDist','LongInt'( 2);
12496:  Function ovPadL( S : string; ch : char; L : integer) : string
12497: end;
12498:
12499:   TFormatSettings = record
12500:     CurrencyFormat: Byte;
12501:     NegCurrFormat: Byte;
12502:     ThousandSeparator: Char;
12503:     DecimalSeparator: Char;
12504:     CurrencyDecimals: Byte;
12505:     DateSeparator: Char;
12506:     TimeSeparator: Char;
12507:     ListSeparator: Char;
12508:     CurrencyString: string;
12509:     ShortDateFormat: string;
12510:     LongDateFormat: string;
12511:     TimeAMString: string;
12512:     TimePMString: string;
12513:     ShortTimeFormat: string;
12514:     LongTimeFormat: string;
12515:     ShortMonthNames: array[1..12] of string;
12516:     LongMonthNames: array[1..12] of string;
12517:     ShortDayNames: array[1..7] of string;
12518:     LongDayNames: array[1..7] of string;
12519:     TwoDigitYearCenturyWindow: Word;
12520:    end;
12521:
12522: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12523: begin
12524:  Function ovFormatSettings : TFormatSettings
12525: end;
12526:
12527: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12528: begin
12529:   TOvcCharSet', 'set of Char
12530:   ovBTable', 'array[0..255] of Byte
12531:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
       HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12532:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12533:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12534:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12535:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12536:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12537:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12538:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12539:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12540:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12541:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12542:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12543:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12544:  Function LoCaseChar( C : Char) : Char
12545:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12546:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12547:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12548:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12549:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12550:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12551:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12552:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12553:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12554:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12555:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12556:  Procedure TrimAllSpacesPChar( P : PChar)
12557:  Function TrimEmbeddedZeros( const S : string) : string
12558:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12559:  Function TrimTrailPrimPChar( S : PChar) : PChar
12560:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12561:  Function TrimTrailingZeros( const S : string) : string
```

```
12562:  Procedure TrimTrailingZerosPChar( P : PChar)
12563:  Function UpCaseChar( C : Char) : Char
12564:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12565:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12566:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12567: end;
12568:
12569: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12570: begin
12571:   //PRaiseFrame', '^TRaiseFrame // will not work
12572:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12573:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12574:  Procedure SafeCloseHandle( var Handle : THandle)
12575:  Procedure ExchangeInteger( X1, X2 : Integer)
12576:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12577:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12578:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12579:
12580: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12581:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12582: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12583:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12584:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12585:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12586:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12587:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12588:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12589:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12590:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12591:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12592:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12593:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12594:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12595:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12596:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12597:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12598:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12599:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12600:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12601:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12602:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12603:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12604:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12605:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12606:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12607:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12608:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12609:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12610:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12611:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12612:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12613:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12614:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12615:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12616:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12617:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12618:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12619:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12620:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12621:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12622:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12623:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12624:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12625:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12626:       var lpLuid: TLargeInteger): BOOL; stdcall;
12627:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12628:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12629:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12630:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12631:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12632:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12633:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12634:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12635:       var GenerateOnClose: BOOL): BOOL; stdcall;
12636:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12637:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12638:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12639:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12640:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12641:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12642:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12643:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12644:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12645:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12646:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12647:       var phkResult: HKEY): Longint; stdcall;
12648:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12649:       var phkResult: HKEY): Longint; stdcall;
12650:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
```

```
12651:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12652:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12653:       lpdwDisposition: PDWORD): Longint; stdcall;
12654:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12655:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12656:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12657:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12658:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12659:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12660:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12661:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12662:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12663:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12664:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12665:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12666:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12667:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12668:       lpcbClass: PDWORD; lpReserved: Pointer;
12669:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12670:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12671:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12672:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12673:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12674:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12675:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12676:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12677:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12678:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12679:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12680:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12681:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12682:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12683:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12684:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12685:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12686:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12687:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12688:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12689:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12690:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12691:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12692:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12693:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12694:
12695:  Function wAddAtom( lpString : PKOLChar) : ATOM
12696:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12697:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12698:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12699:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
      lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12700:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12701:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12702:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12703:  Function wCreateDirectoryEx(lpTemplateDirectory,
      lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12704:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
      bInitialState:BOOL;lpName:PKOLChar):THandle;
12705:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
      PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12706:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
      dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12707:  Function wCreateHardLink(lpFileName,
      lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12708:  Function
      CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12709:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
      nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12710:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
      lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
      Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
      lpProcessInfo:TProcessInformation):BOOL
12711:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
      Longint; lpName : PKOLChar) : THandle
12712:  Function
      wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12713:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12714:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12715:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12716:  //Function
      wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12717:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12718:  //Function
      wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12719:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12720:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12721:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12722:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
```

```
12723:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12724:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12725:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
        dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12726:  Function wFindAtom( lpString : PKOLChar) : ATOM
12727:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12728:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12729:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12730:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12731:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12732:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12733:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12734:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12735:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12736:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12737:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12738:  Function wGetCommandLine : PKOLChar
12739:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12740:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12741:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12742:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12743:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12744:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12745:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12746:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12747:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12748:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12749:  Function wGetEnvironmentStrings : PKOLChar
12750:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12751:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12752:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12753:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12754:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12755:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12756:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12757:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12758:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12759:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12760:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12761:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12762:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12763:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
        nSize:DWORD; lpFileName : PKOLChar) : DWORD
12764:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12765:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12766:  Function wGetProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12767:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12768:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12769: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
        lpCharType):BOOL
12770:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12771:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12772:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12773: //Function
        wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12774: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12775: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
        : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
        lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12776:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12777:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12778:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12779:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12780:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12781:  Function
        wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12782:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12783:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12784:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12785:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12786:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
        TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12787:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12788:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12789:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
```

```
12790:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12791:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12792:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12793:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
        lpNumberOfEventsRead:DWORD):BOOL;
12794:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12795:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12796:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
        lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12797:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
        lpNumbOfEventsRead:DWORD):BOOL;
12798:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
        : TCoord; var lpReadRegion : TSmallRect) : BOOL
12799:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12800:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12801:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
        lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12802:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12803:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12804:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12805:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12806:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12807:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12808:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12809:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12810:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12811:  //Function wUpdateResource(hUpdate:THandle;lpType,
        lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12812:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12813:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12814:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
        DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12815:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
        var lpNumberOfEventsWritten : DWORD) : BOOL
12816:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
        TCoord; var lpWriteRegion : TSmallRect) : BOOL
12817:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12818:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12819:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12820:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12821:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12822:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12823:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12824:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12825:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12826:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12827:  Function wlstrlen( lpString : PKOLChar) : Integer
12828:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
        PNetConnectInfoStruct) : DWORD
12829:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12830:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12831:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12832:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12833:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12834:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12835:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12836:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12837:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12838:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
        : PKOLChar; nNameBufSize : DWORD) : DWORD
12839:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12840:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12841:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12842: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
12843:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12844: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12845: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12846:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12847:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12848:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12849:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12850:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12851: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12852: //Func wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12853: //Func wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12854:  Function wAddFontResource( FileName : PKOLChar) : Integer
12855:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
```

```
12856:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12857:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12858:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12859:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12860: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12861:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
       fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
       fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12862:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12863:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12864: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12865:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12866:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12867: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
       pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12868: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12869:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12870:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12871:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12872:  //Function wExtTextOut(DC:HDC;X,
       Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12873:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12874:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12875:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12876:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12877: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12878: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12879:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12880:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12881: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12882: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
       lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12883:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12884: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12885:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12886: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12887:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12888:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12889:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12890:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12891:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12892:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12893:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12894:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12895:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12896:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12897:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12898:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12899:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12900:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12901:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12902:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12903:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12904:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12905:  //Function
       wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12906:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12907: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
       dwFlags : DWORD; lParam : Pointer) : Longint
12908:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12909:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12910:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12911:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12912:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12913:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12914: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12915:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12916:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12917:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12918:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12919:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12920:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12921:  //Function wCreateDesktop(lpszDesktop,
       lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12922:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
       HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12923:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
       lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12924:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
       hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12925: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
       nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12926:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
       dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12927:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12928:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12929:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12930:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12931:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
       : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
```

```
12932:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
        : TFNDlgProc; dwInitParam : LPARAM) : Integer
12933:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12934:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
        nIDStaticPath:Integer;uFileType:UINT):Integer;
12935:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
        nIDStaticPath:Int;uFiletype:UINT):Int;
12936:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12937:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12938: //FuncwDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
        cy:Int;Flags:UINT):BOOL;
12939:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12940:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12941:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12942:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
        pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12943: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12944:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12945:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12946:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12947:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12948:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12949:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12950:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12951:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12952:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12953:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12954:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12955:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
        lpnTabStopPositions) : DWORD
12956: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
        lpnLengthNeed:DWORD)BOOL;
12957:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12958:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12959:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12960:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12961: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
        nHeigt:Int):BOOL;
12962:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12963:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12964:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12965:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12966:  Function wIsCharLower( ch : KOLChar) : BOOL
12967:  Function wIsCharUpper( ch : KOLChar) : BOOL
12968:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12969:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12970:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12971:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12972:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12973:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12974:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12975:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12976:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12977:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12978:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12979:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12980:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
12981:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
12982:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12983:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
12984:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12985:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
12986:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
12987:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12988:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12989:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12990:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
12991:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
12992:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12993:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12994:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
12995: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
12996: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
12997:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
12998: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12999:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
13000:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
13001:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13002:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
13003:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
        lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
13004:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
        lpdwResult:DWORD): LRESULT
13005:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
13006:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
13007:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
13008:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
13009:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
13010: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
```

```
13011:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
13012:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
13013:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
13014:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
13015: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
13016:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
      lpnTabStopPositions,nTabOrigin:Int):Longint;
13017:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
13018:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
13019:  Function wVkKeyScan( ch : KOLChar) : SHORT
13020:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
13021:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
13022:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
13023:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
13024:
13025: //TestDrive!
13026: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
13027:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
13028:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
13029:  Function GetLocalUserSidStr( const UserName : string) : string
13030:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
13031:  Function Impersonate2User( const domain : string; const user : string) : boolean
13032:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
13033:  Function KillProcessbyname( const exename : string; var found : integer) : integer
13034:  Function getWinProcessList : TStringList
13035:  function WaitTilClose(hWnd: Integer): Integer;
13036:  function DoUserMsgs: Boolean;
13037:  function MsgFunc(hWnd,Msg,wParam,lParam:Integer):Integer; stdcall;
13038:  procedure ShowMsg(hParent: Integer; const Mess, Title: String);  //modal but NOT blockable
13039:  procedure DeleteMsgForm(Handle: Integer);
13040:  procedure DisableForms;
13041:  function FoundTopLevel(hWnd, LParam: Integer): BOOL; StdCall;
13042: end;
13043:
13044: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
13045: begin
13046:  'AfMaxSyncSlots','LongInt'( 64);
13047:  'AfSynchronizeTimeout','LongInt'( 2000);
13048:   TAfSyncSlotID', 'DWORD
13049:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
13050:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
13051:   TAfSafeDirectSyncEvent', 'Procedure
13052:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
13053:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
13054:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
13055:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
13056:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
13057:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
13058:  Function AfIsSyncMethod : Boolean
13059:  Function AfSyncWnd : HWnd
13060:  Function AfSyncStatistics : TAfSyncStatistics
13061:  Procedure AfClearSyncStatistics
13062: end;
13063:
13064: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
13065: begin
13066:  'fBinary','LongWord')( $00000001);
13067:  'fParity','LongWord'( $00000002);
13068:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
13069:  'fOutxDsrFlow','LongWord')( $00000008);
13070:  'fDtrControl','LongWord')( $00000030);
13071:  'fDtrControlDisable','LongWord')( $00000000);
13072:  'fDtrControlEnable','LongWord')( $00000010);
13073:  'fDtrControlHandshake','LongWord')( $00000020);
13074:  'fDsrSensitivity','LongWord')( $00000040);
13075:  'fTXContinueOnXoff','LongWord')( $00000080);
13076:  'fOutX','LongWord')( $00000100);
13077:  'fInX','LongWord')( $00000200);
13078:  'fErrorChar','LongWord')( $00000400);
13079:  'fNull','LongWord')( $00000800);
13080:  'fRtsControl','LongWord')( $00003000);
13081:  'fRtsControlDisable','LongWord')( $00000000);
13082:  'fRtsControlEnable','LongWord')( $00001000);
13083:  'fRtsControlHandshake','LongWord')( $00002000);
13084:  'fRtsControlToggle','LongWord')( $00003000);
13085:  'fAbortOnError','LongWord')( $00004000);
13086:  'fDummy2','LongWord')( $FFFF8000);
13087:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13088:   FindClass('TOBJECT'),'EAfComPortCoreError
13089:   FindClass('TOBJECT'),'TAfComPortCore
13090:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13091:    +'tKind : TAfCoreEvent; Data : DWORD)
13092:   SIRegister_TAfComPortCoreThread(CL);
13093:   SIRegister_TAfComPortEventThread(CL);
13094:   SIRegister_TAfComPortWriteThread(CL);
13095:   SIRegister_TAfComPortCore(CL);
13096:  Function FormatDeviceName( PortNumber : Integer) : string
13097: end;
13098:
```

```
13099: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13100: begin
13101:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13102:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13103:   SIRegister_TApplicationFileIO(CL);
13104:   TDataFileCapability', '( dfcRead, dfcWrite )
13105:   TDataFileCapabilities', 'set of TDataFileCapability
13106:   SIRegister_TDataFile(CL);
13107:   //TDataFileClass', 'class of TDataFile
13108:  Function ApplicationFileIODefined : Boolean
13109:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13110:  Function FileStreamExists(const fileName: String) : Boolean
13111:  //Procedure Register
13112: end;
13113:
13114: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13115: begin
13116:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13117:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13118:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13119:   TALFBXScale', 'Integer
13120:   FindClass('TOBJECT'),'EALFBXConvertError
13121:   SIRegister_EALFBXError(CL);
13122:   SIRegister_EALFBXException(CL);
13123:   FindClass('TOBJECT'),'EALFBXGFixError
13124:   FindClass('TOBJECT'),'EALFBXDSQLError
13125:   FindClass('TOBJECT'),'EALFBXDynError
13126:   FindClass('TOBJECT'),'EALFBXGBakError
13127:   FindClass('TOBJECT'),'EALFBXGSecError
13128:   FindClass('TOBJECT'),'EALFBXLicenseError
13129:   FindClass('TOBJECT'),'EALFBXGStatError
13130:   //EALFBXExceptionClass', 'class of EALFBXError
13131:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13132:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13133:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13134:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13135:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13136:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13137:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13138:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13139:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13140:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13141:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13142:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13143:   TALFBXTransParams', 'set of TALFBXTransParam
13144:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13145:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13146:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13147:  'cALFBXMaxParamLength','LongInt'( 125);
13148:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13149:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13150:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13151:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13152:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13153:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13154:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13155:   SIRegister_TALFBXSQLDA(CL);
13156:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13157:   SIRegister_TALFBXPoolStream(CL);
13158:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13159:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13160:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13161:   //TALFBXArrayDesc', 'TISCArrayDesc
13162:   //TALFBXBlobDesc', 'TISCBlobDesc
13163:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13164:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13165:   SIRegister_TALFBXSQLResult(CL);
13166:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13167:   SIRegister_TALFBXSQLParams(CL);
13168:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13169:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13170:    +'atementType : TALFBXStatementType; end
13171:   FindClass('TOBJECT'),'TALFBXLibrary
13172:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13173:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13174:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13175:    //+' Excep : EALFBXExceptionClass)
13176:   SIRegister_TALFBXLibrary(CL);
13177:  'cAlFBXDateOffset','LongInt'( 15018);
13178:  'cALFBXTimeCoeff','LongInt'( 864000000);
13179:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13180:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13181:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13182:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13183:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13184:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13185:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13186:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13187:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
```

```
13188:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13189:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13190:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13191:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13192:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13193: end;
13194:
13195: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13196: begin
13197:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13198:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13199:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13200:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13201:    +'teger; First : Integer; CacheThreshold : Integer; end
13202:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13203:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13204:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13205:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13206:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13207:   SIRegister_TALFBXClient(CL);
13208:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13209:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13210:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13211:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13212:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13213:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13214:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13215:   SIRegister_TALFBXConnectionPoolClient(CL);
13216:   SIRegister_TALFBXEventThread(CL);
13217:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13218: end;
13219:
13220: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13221: begin
13222: _OSVERSIONINFOA = record
13223:     dwOSVersionInfoSize: DWORD;
13224:     dwMajorVersion: DWORD;
13225:     dwMinorVersion: DWORD;
13226:     dwBuildNumber: DWORD;
13227:     dwPlatformId: DWORD;
13228:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13229:   end;
13230:  TOSVersionInfoA', '_OSVERSIONINFOA
13231:   TOSVersionInfo', 'TOSVersionInfoA
13232:  'WS_EX_RIGHT','LongWord')( $00001000);
13233:  'WS_EX_LEFT','LongWord')( $00000000);
13234:  'WS_EX_RTLREADING','LongWord')( $00002000);
13235:  'WS_EX_LTRREADING','LongWord')( $00000000);
13236:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13237:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13238:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13239:  'LAYOUT_RTL','LongWord')( $00000001);
13240:  'LAYOUT_BTT','LongWord')( $00000002);
13241:  'LAYOUT_VBH','LongWord')( $00000004);
13242:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13243:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13244:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13245:  Function GetLayout( dc : hdc) : DWORD
13246:  Function IsBidi : Boolean
13247:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13248:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13249:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13250:  Function GetPriorityClass( hProcess : THandle) : DWORD
13251:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13252:  Function CloseClipboard : BOOL
13253:  Function GetClipboardSequenceNumber : DWORD
13254:  Function GetClipboardOwner : HWND
13255:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13256:  Function GetClipboardViewer : HWND
13257:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13258:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13259:  Function GetClipboardData( uFormat : UINT) : THandle
13260:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13261:  Function CountClipboardFormats : Integer
13262:  Function EnumClipboardFormats( format : UINT) : UINT
13263:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13264:  Function EmptyClipboard : BOOL
13265:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13266:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13267:  Function GetOpenClipboardWindow : HWND
13268:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13269:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13270:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13271:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13272:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13273:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13274:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13275:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13276:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
```

```
13277: end;
13278:
13279: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13280: begin
13281:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13282:  Function GetTemporaryFilesPath : String
13283:  Function GetTemporaryFileName : String
13284:  Function FindFileInPaths( const fileName, paths : String) : String
13285:  Function PathsToString( const paths : TStrings) : String
13286:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13287:  //Function MacroExpandPath( const aPath : String) : String
13288: end;
13289:
13290: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13291: begin
13292:   SIRegister_TALMultiPartBaseContent(CL);
13293:   SIRegister_TALMultiPartBaseContents(CL);
13294:   SIRegister_TAlMultiPartBaseStream(CL);
13295:   SIRegister_TALMultipartBaseEncoder(CL);
13296:   SIRegister_TALMultipartBaseDecoder(CL);
13297:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13298:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13299:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13300: end;
13301:
13302: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13303: begin
13304:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13305:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13306:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13307:  Function aAllocPadedMem( Size : Cardinal) : TObject
13308:  Procedure aFreePadedMem( var P : TObject);
13309:  Procedure aFreePadedMem1( var P : PChar);
13310:  Function aCheckPadedMem( P : Pointer) : Byte
13311:  Function aGetPadMemSize( P : Pointer) : Cardinal
13312:  Function aAllocMem( Size : Cardinal) : Pointer
13313:  Function aStrLen( const Str : PChar) : Cardinal
13314:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13315:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13316:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13317:  Function aStrEnd( const Str : PChar) : PChar
13318:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13319:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13320:  Function aPCharLength( const Str : PChar) : Cardinal
13321:  Function aPCharUpper( Str : PChar) : PChar
13322:  Function aPCharLower( Str : PChar) : PChar
13323:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13324:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13325:  Function aCopyTail( const S : String; Len : Integer) : String
13326:  Function aInt2Thos( I : Int64) : String
13327:  Function aUpperCase( const S : String) : String
13328:  Function aLowerCase( const S : string) : String
13329:  Function aCompareText( const S1, S2 : string) : Integer
13330:  Function aSameText( const S1, S2 : string) : Boolean
13331:  Function aInt2Str( Value : Int64) : String
13332:  Function aStr2Int( const Value : String) : Int64
13333:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13334:  Function aGetFileExt( const FileName : String) : String
13335:  Function aGetFilePath( const FileName : String) : String
13336:  Function aGetFileName( const FileName : String) : String
13337:  Function aChangeExt( const FileName, Extension : String) : String
13338:  Function aAdjustLineBreaks( const S : string) : string
13339:  Function aGetWindowStr( WinHandle : HWND) : String
13340:  Function aDiskSpace( Drive : String) : TdriveSize
13341:  Function aFileExists( FileName : String) : Boolean
13342:  Function aFileSize( FileName : String) : Int64
13343:  Function aDirectoryExists( const Name : string) : Boolean
13344:  Function aSysErrorMessage( ErrorCode : Integer) : string
13345:  Function aShortPathName( const LongName : string) : string
13346:  Function aGetWindowVer : TWinVerRec
13347:  procedure InitDriveSpacePtr;
13348: end;
13349:
13350: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13351: begin
13352:  aZero','LongInt'( 0);
13353:  'makeappDEF','LongInt'( - 1);
13354:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13355:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13356:  'CS_KEYCVTWINDOW','LongInt'( 4);
13357:  'CS_DBLCLKS','LongInt'( 8);
13358:  'CS_OWNDC','LongWord')( $20);
13359:  'CS_CLASSDC','LongWord')( $40);
13360:  'CS_PARENTDC','LongWord')( $80);
13361:  'CS_NOKEYCVT','LongWord')( $100);
13362:  'CS_NOCLOSE','LongWord')( $200);
13363:  'CS_SAVEBITS','LongWord')( $800);
13364:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13365:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
```

```
13366:  'CS_GLOBALCLASS','LongWord')( $4000);
13367:  'CS_IME','LongWord')( $10000);
13368:  'CS_DROPSHADOW','LongWord')( $20000);
13369:   //PPanelFunc', '^TPanelFunc // will not work
13370:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13371:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13372:   TFontLooks', 'set of TFontLook
13373:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13374:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13375:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13376:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13377:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13378:  Procedure RunMsgLoop( Show : Boolean)
13379:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13380:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
       ID_Number:Cardinal;hFont:Int):Int;
13381:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13382:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13383:  Function MakePanel(Left,Top,Width,Height,
       hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13384:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13385:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13386:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13387: end;
13388:
13389: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13390: begin
13391:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13392:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13393:   TScreenSaverOptions', 'set of TScreenSaverOption
13394:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13395:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13396:   SIRegister_TScreenSaver(CL);
13397:  //Procedure Register
13398:  Procedure SetScreenSaverPassword
13399: end;
13400:
13401: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13402: begin
13403:   FindClass('TOBJECT'),'TXCollection
13404:   SIRegister_EFilerException(CL);
13405:   SIRegister_TXCollectionItem(CL);
13406:   //TXCollectionItemClass', 'class of TXCollectionItem
13407:   SIRegister_TXCollection(CL);
13408:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13409:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13410:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13411:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13412:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13413:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13414: end;
13415:
13416: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13417: begin
13418:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13419:  Procedure xglMapTexCoordToNull
13420:  Procedure xglMapTexCoordToMain
13421:  Procedure xglMapTexCoordToSecond
13422:  Procedure xglMapTexCoordToDual
13423:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13424:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13425:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13426:  Procedure xglBeginUpdate
13427:  Procedure xglEndUpdate
13428:  Procedure xglPushState
13429:  Procedure xglPopState
13430:  Procedure xglForbidSecondTextureUnit
13431:  Procedure xglAllowSecondTextureUnit
13432:  Function xglGetBitWiseMapping : Cardinal
13433: end;
13434:
13435: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13436: begin
13437:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13438:   TBaseListOptions', 'set of TBaseListOption
13439:   SIRegister_TBaseList(CL);
13440:   SIRegister_TBaseVectorList(CL);
13441:   SIRegister_TAffineVectorList(CL);
13442:   SIRegister_TVectorList(CL);
13443:   SIRegister_TTexPointList(CL);
13444:   SIRegister_TXIntegerList(CL);
13445:   //PSingleArrayList', '^TSingleArrayList // will not work
13446:   SIRegister_TSingleList(CL);
13447:   SIRegister_TByteList(CL);
13448:   SIRegister_TQuaternionList(CL);
13449:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13450:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13451:  Procedure FastQuickSortLists(startIndex,
       endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
```

```
13452: end;
13453:
13454: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13455: begin
13456:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13457:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13458:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
       indices:TIntegerList;list:TAffineVectorList);
13459:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
       indices:TIntegerList;list:TAffineVectorList);
13460:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
       normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13461:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13462:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13463:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13464:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13465:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13466:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13467:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13468:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13469:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13470:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
       edgesTriangles : TIntegerList) : TIntegerList
13471:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13472:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
       TPersistentObjectList;
13473:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13474:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
       TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13475: end;
13476:
13477: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13478: begin
13479:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13480:  Procedure FreeMemAndNil( var P : TObject)
13481:  Function PCharOrNil( const S : string) : PChar
13482:   SIRegister_TJclReferenceMemoryStream(CL);
13483:   FindClass('TOBJECT'),'EJclVMTError
13484:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13485:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13486:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13487:   PDynamicIndexList', '^TDynamicIndexList // will not work
13488:   PDynamicAddressList', '^TDynamicAddressList // will not work
13489:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13490:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13491:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13492:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13493:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13494:  Function GetInitTable( AClass : TClass) : PTypeInfo
13495:   PFieldEntry', '^TFieldEntry // will not work}
13496:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13497:  Function JIsClass( Address : Pointer) : Boolean
13498:  Function JIsObject( Address : Pointer) : Boolean
13499:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13500:   TDigitCount', 'Integer
13501:   SIRegister_TJclNumericFormat(CL);
13502:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13503:   TTextHandler', 'Procedure ( const Text : string)
13504: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13505:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13506:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13507:  Function ReadKey : Char   //to and from the DOS console !
13508:   TModuleHandle', 'HINST
13509:   //TModuleHandle', 'Pointer
13510:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13511:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13512:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13513:  Procedure UnloadModule( var Module : TModuleHandle)
13514:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13515:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13516:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13517:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13518:   FindClass('TOBJECT'),'EJclConversionError
13519:  Function JStrToBoolean( const S : string) : Boolean
13520:  Function JBooleanToStr( B : Boolean) : string
13521:  Function JIntToBool( I : Integer) : Boolean
13522:  Function JBoolToInt( B : Boolean) : Integer
13523:  'ListSeparator','String ';
13524:  'ListSeparator1','String ':
13525:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13526:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13527:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13528:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13529:  Function ListItemCount( const List, Separator : string) : Integer
13530:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13531:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13532:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13533:  Function SystemTObjectInstance : LongWord
```

```
13534:  Function IsCompiledWithPackages : Boolean
13535:  Function JJclGUIDToString( const GUID : TGUID) : string
13536:  Function JJclStringToGUID( const S : string) : TGUID
13537:   SIRegister_TJclIntfCriticalSection(CL);
13538:   SIRegister_TJclSimpleLog(CL);
13539:  Procedure InitSimpleLog( const ALogFileName : string)
13540: end;
13541:
13542: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13543: begin
13544:   FindClass('TOBJECT'),'EJclBorRADException
13545:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13546:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13547:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13548:   TJclBorRADToolPath', 'string
13549:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13550:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13551:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13552:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13553:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13554:  BorRADToolRepositoryFormsPage','String 'Forms
13555:  BorRADToolRepositoryProjectsPage','String 'Projects
13556:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13557:  BorRADToolRepositoryObjectType','String 'Type
13558:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13559:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13560:  BorRADToolRepositoryObjectName','String 'Name
13561:  BorRADToolRepositoryObjectPage','String 'Page
13562:  BorRADToolRepositoryObjectIcon','String 'Icon
13563:  BorRADToolRepositoryObjectDescr','String 'Description
13564:  BorRADToolRepositoryObjectAuthor','String 'Author
13565:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13566:  BorRADToolRepositoryObjectDesigner','String 'Designer
13567:  BorRADToolRepositoryDesignerDfm','String 'dfm
13568:  BorRADToolRepositoryDesignerXfm','String 'xfm
13569:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13570:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13571:  SourceExtensionDelphiPackage','String '.dpk
13572:  SourceExtensionBCBPackage','String '.bpk
13573:  SourceExtensionDelphiProject','String '.dpr
13574:  SourceExtensionBCBProject','String '.bpr
13575:  SourceExtensionBDSProject','String '.bdsproj
13576:  SourceExtensionDProject','String '.dproj
13577:  BinaryExtensionPackage','String '.bpl
13578:  BinaryExtensionLibrary','String '.dll
13579:  BinaryExtensionExecutable','String '.exe
13580:  CompilerExtensionDCP','String '.dcp
13581:  CompilerExtensionBPI','String '.bpi
13582:  CompilerExtensionLIB','String '.lib
13583:  CompilerExtensionTDS','String '.tds
13584:  CompilerExtensionMAP','String '.map
13585:  CompilerExtensionDRC','String '.drc
13586:  CompilerExtensionDEF','String '.def
13587:  SourceExtensionCPP','String '.cpp
13588:  SourceExtensionH','String '.h
13589:  SourceExtensionPAS','String '.pas
13590:  SourceExtensionDFM','String '.dfm
13591:  SourceExtensionXFM','String '.xfm
13592:  SourceDescriptionPAS','String 'Pascal source file
13593:  SourceDescriptionCPP','String 'C++ source file
13594:  DesignerVCL','String 'VCL
13595:  DesignerCLX','String 'CLX
13596:  ProjectTypePackage','String 'package
13597:  ProjectTypeLibrary','String 'library
13598:  ProjectTypeProgram','String 'program
13599:  Personality32Bit','String '32 bit
13600:  Personality64Bit','String '64 bit
13601:  PersonalityDelphi','String 'Delphi
13602:  PersonalityDelphiDotNet','String 'Delphi.net
13603:  PersonalityBCB','String 'C++Builder
13604:  PersonalityCSB','String 'C#Builder
13605:  PersonalityVB','String 'Visual Basic
13606:  PersonalityDesign','String 'Design
13607:  PersonalityUnknown','String 'Unknown personality
13608:  PersonalityBDS','String 'Borland Developer Studio
13609:  DOFDirectoriesSection','String 'Directories
13610:  DOFUnitOutputDirKey','String 'UnitOutputDir
13611:  DOFSearchPathName','String 'SearchPath
13612:  DOFConditionals','String 'Conditionals
13613:  DOFLinkerSection','String 'Linker
13614:  DOFPackagesKey','String 'Packages
13615:  DOFCompilerSection','String 'Compiler
13616:  DOFPackageNoLinkKey','String 'PackageNoLink
13617:  DOFAdditionalSection','String 'Additional
13618:  DOFOptionsKey','String 'Options
13619:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13620:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13621:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13622:   TJclBorPersonalities', 'set of TJclBorPersonality
```

```
13623:   TJclBorDesigner', '( bdVCL, bdCLX )
13624:   TJclBorDesigners', 'set of TJClBorDesigner
13625:   TJclBorPlatform', '( bp32bit, bp64bit )
13626:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13627:   SIRegister_TJclBorRADToolInstallationObject(CL);
13628:   SIRegister_TJclBorlandOpenHelp(CL);
13629:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13630:   TJclHelp2Objects', 'set of TJclHelp2Object
13631:   SIRegister_TJclHelp2Manager(CL);
13632:   SIRegister_TJclBorRADToolIdeTool(CL);
13633:   SIRegister_TJclBorRADToolIdePackages(CL);
13634:   SIRegister_IJclCommandLineTool(CL);
13635:   FindClass('TOBJECT'),'EJclCommandLineToolError
13636:   SIRegister_TJclCommandLineTool(CL);
13637:   SIRegister_TJclBorlandCommandLineTool(CL);
13638:   SIRegister_TJclBCC32(CL);
13639:   SIRegister_TJclDCC32(CL);
13640:   TJclDCC', 'TJclDCC32
13641:   SIRegister_TJclBpr2Mak(CL);
13642:   SIRegister_TJclBorlandMake(CL);
13643:   SIRegister_TJclBorRADToolPalette(CL);
13644:   SIRegister_TJclBorRADToolRepository(CL);
13645:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13646:   TCommandLineTools', 'set of TCommandLineTool
13647:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13648:   SIRegister_TJclBorRADToolInstallation(CL);
13649:   SIRegister_TJclBCBInstallation(CL);
13650:   SIRegister_TJclDelphiInstallation(CL);
13651:   SIRegister_TJclDCCIL(CL);
13652:   SIRegister_TJclBDSInstallation(CL);
13653:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13654:   SIRegister_TJclBorRADToolInstallations(CL);
13655:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13656:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13657:  Function IsDelphiPackage( const FileName : string) : Boolean
13658:  Function IsDelphiProject( const FileName : string) : Boolean
13659:  Function IsBCBPackage( const FileName : string) : Boolean
13660:  Function IsBCBProject( const FileName : string) : Boolean
13661:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13662:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13663:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
      Descript:PString;
13664:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
      Descript:PString
13665:  function SamePath(const Path1, Path2: string): Boolean;
13666: end;
13667:
13668: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13669: begin
13670:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13671:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13672:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13673:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13674:  'LPathSeparator','String '/
13675:  'LDirDelimiter','String '/
13676:  'LDirSeparator','String ':
13677:  'JXPathDevicePrefix','String '\\.\
13678:  'JXPathSeparator','String '\
13679:  'JXDirDelimiter','String '\
13680:  'JXDirSeparator','String ';
13681:  'JXPathUncPrefix','String '\\
13682:  'faNormalFile','LongWord')( $00000080);
13683:  //'faUnixSpecific',' faSymLink);
13684:   JXTCompactPath', '( cpCenter, cpEnd )
13685:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13686:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13687:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13688:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13689:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13690:
13691:  Function jxPathAddSeparator( const Path : string) : string
13692:  Function jxPathAddExtension( const Path, Extension : string) : string
13693:  Function jxPathAppend( const Path, Append : string) : string
13694:  Function jxPathBuildRoot( const Drive : Byte) : string
13695:  Function jxPathCanonicalize( const Path : string) : string
13696:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13697:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13698:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13699:  Function jxPathExtractFileDirFixed( const S : string) : string
13700:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13701:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13702:  Function jxPathGetDepth( const Path : string) : Integer
13703:  Function jxPathGetLongName( const Path : string) : string
13704:  Function jxPathGetShortName( const Path : string) : string
13705:  Function jxPathGetLongName( const Path : string) : string
13706:  Function jxPathGetShortName( const Path : string) : string
13707:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13708:  Function jxPathGetTempPath : string
13709:  Function jxPathIsAbsolute( const Path : string) : Boolean
```

```
13710:  Function jxPathIsChild( const Path, Base : string) : Boolean
13711:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13712:  Function jxPathIsUNC( const Path : string) : Boolean
13713:  Function jxPathRemoveSeparator( const Path : string) : string
13714:  Function jxPathRemoveExtension( const Path : string) : string
13715:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13716:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13717:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13718:   JxTFileListOptions', 'set of TFileListOption
13719:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13720:   TFileHandler', 'Procedure ( const FileName : string)
13721:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13722:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13723:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
       AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
       FileMatchFunc:TFileMatchFunc):Bool;
13724:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13725:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13726:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13727:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13728:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
       RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13729:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
       IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13730:  Procedure jxCreateEmptyFile( const FileName : string)
13731:  Function jxCloseVolume( var Volume : THandle) : Boolean
13732:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13733:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13734:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13735:  Function jxDelTree( const Path : string) : Boolean
13736:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13737:  Function jxDiskInDrive( Drive : Char) : Boolean
13738:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13739:  Function jxFileCreateTemp( var Prefix : string) : THandle
13740:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13741:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13742:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13743:  Function jxFileExists( const FileName : string) : Boolean
13744:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13745:  Function jxFileRestore( const FileName : string) : Boolean
13746:  Function jxGetBackupFileName( const FileName : string) : string
13747:  Function jxIsBackupFileName( const FileName : string) : Boolean
13748:  Function jxFileGetDisplayName( const FileName : string) : string
13749:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13750:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13751:  Function jxFileGetSize( const FileName : string) : Int64
13752:  Function jxFileGetTempName( const Prefix : string) : string
13753:  Function jxFileGetTypeName( const FileName : string) : string
13754:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13755:  Function jxForceDirectories( Name : string) : Boolean
13756:  Function jxGetDirectorySize( const Path : string) : Int64
13757:  Function jxGetDriveTypeStr( const Drive : Char) : string
13758:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13759:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13760:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13761:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13762:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13763:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
       ResolveSymLinks:Boolean):Integer
13764:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13765:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13766:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13767:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13768:  Function jxGetFileCreation( const FName : string) : TFileTime;
13769:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13770:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13771:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13772:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13773:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13774:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13775:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13776:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13777:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13778:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13779:  Function jxGetModulePath( const Module : HMODULE) : string
13780:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13781:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13782:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13783:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13784:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13785:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13786:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13787:  Function jxOpenVolume( const Drive : Char) : THandle
13788:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13789:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13790:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13791:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13792:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13793:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
```

```
13794:  Procedure jxShredFile( const FileName : string; Times : Integer)
13795:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13796:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13797:  Function jxSymbolicLinkTarget( const Name : string) : string
13798:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13799:   SIRegister_TJclCustomFileAttrMask(CL);
13800:   SIRegister_TJclFileAttributeMask(CL);
13801:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13802:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13803:   TFileSearchOptions', 'set of TFileSearchOption
13804:   TFileSearchTaskID', 'Integer
13805:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13806:    +'hTaskID; const Aborted : Boolean)
13807:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13808:   SIRegister_IJclFileEnumerator(CL);
13809:   SIRegister_TJclFileEnumerator(CL);
13810:  Function JxFileSearch : IJclFileEnumerator
13811:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13812:   JxTFileFlags', 'set of TFileFlag
13813:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13814:   SIRegister_TJclFileVersionInfo(CL);
13815:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13816:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13817:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13818:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13819:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13820:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13821:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13822:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13823:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13824:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13825:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13826:   SIRegister_TJclTempFileStream(CL);
13827:   FindClass('TOBJECT'),'TJclCustomFileMapping
13828:   SIRegister_TJclFileMappingView(CL);
13829:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13830:   SIRegister_TJclCustomFileMapping(CL);
13831:   SIRegister_TJclFileMapping(CL);
13832:   SIRegister_TJclSwapFileMapping(CL);
13833:   SIRegister_TJclFileMappingStream(CL);
13834:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13835:   //PPCharArray', '^TPCharArray // will not work
13836:   SIRegister_TJclMappedTextReader(CL);
13837:   SIRegister_TJclFileMaskComparator(CL);
13838:   FindClass('TOBJECT'),'EJclPathError
13839:   FindClass('TOBJECT'),'EJclFileUtilsError
13840:   FindClass('TOBJECT'),'EJclTempFileStreamError
13841:   FindClass('TOBJECT'),'EJclTempFileStreamError
13842:   FindClass('TOBJECT'),'EJclFileMappingError
13843:   FindClass('TOBJECT'),'EJclFileMappingViewError
13844:  Function jxPathGetLongName2( const Path : string) : string
13845:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13846:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13847:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13848:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13849:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13850:  Procedure jxPathListAddItems( var List : string; const Items : string)
13851:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13852:  Procedure jxPathListDelItems( var List : string; const Items : string)
13853:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13854:  Function jxPathListItemCount( const List : string) : Integer
13855:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13856:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13857:  Function jxPathListItemIndex( const List, Item : string) : Integer
13858:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13859:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13860:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13861:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string) : Integer
13862: end;
13863:
13864: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13865: begin
13866:  'UTF8FileHeader','String #$ef#$bb#$bf);
13867:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13868:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13869:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13870:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13871:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13872:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13873:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13874:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13875:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13876:  Function lFileIsReadable( const AFilename : string) : boolean
13877:  Function lFileIsWritable( const AFilename : string) : boolean
13878:  Function lFileIsText( const AFilename : string) : boolean
```

```
13879:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13880:  Function lFileIsExecutable( const AFilename : string) : boolean
13881:  Function lFileIsSymlink( const AFilename : string) : boolean
13882:  Function lFileIsHardLink( const AFilename : string) : boolean
13883:  Function lFileSize( const Filename : string) : int64;
13884:  Function lGetFileDescription( const AFilename : string) : string
13885:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13886:  Function lTryReadAllLinks( const Filename : string) : string
13887:  Function lDirPathExists( const FileName : String) : Boolean
13888:  Function lForceDirectory( DirectoryName : string) : boolean
13889:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13890:  Function lProgramDirectory : string
13891:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13892:  Function lExtractFileNameOnly( const AFilename : string) : string
13893:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13894:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13895:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13896:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13897:  Function lAppendPathDelim( const Path : string) : string
13898:  Function lChompPathDelim( const Path : string) : string
13899:  Function lTrimFilename( const AFilename : string) : string
13900:  Function lCleanAndExpandFilename( const Filename : string) : string
13901:  Function lCleanAndExpandDirectory( const Filename : string) : string
13902:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13903:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
      AlwaysRequireSharedBaseFolder : Boolean) : string
13904:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13905:  Function lFileIsInPath( const Filename, Path : string) : boolean
13906:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13907:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13908:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13909:  'AllDirectoryEntriesMask','String '*
13910:  Function lGetAllFilesMask : string
13911:  Function lGetExeExt : string
13912:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
      TSearchFileInPathFlags) : string
13913:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
      TSearchFileInPathFlags) : TStrings
13914:  Function lFindDiskFilename( const Filename : string) : string
13915:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13916:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13917:  Function lGetDarwinSystemFilename( Filename : string) : string
13918:   SIRegister_TFileIterator(CL);
13919:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13920:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13921:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13922:   SIRegister_TFileSearcher(CL);
13923:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13924:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13925:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13926:  // TCopyFileFlags', 'set of TCopyFileFlag
13927:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13928:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13929:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13930:  Function lReadFileToString( const Filename : string) : string
13931:  Function lGetTempFilename( const Directory, Prefix : string) : string
13932:  {Function NeedRTLAnsi : boolean
13933:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13934:  Function UTF8ToSys( const s : string) : string
13935:  Function SysToUTF8( const s : string) : string
13936:  Function ConsoleToUTF8( const s : string) : string
13937:  Function UTF8ToConsole( const s : string) : string}
13938:  Function FileExistsUTF8( const Filename : string) : boolean
13939:  Function FileAgeUTF8( const FileName : string) : Longint
13940:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13941:  Function ExpandFileNameUTF8( const FileName : string) : string
13942:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13943:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13944:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13945:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13946:  Procedure FindCloseUTF8( var F : TSearchrec)
13947:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13948:  Function FileGetAttrUTF8( const FileName : String) : Longint
13949:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13950:  Function DeleteFileUTF8( const FileName : String) : Boolean
13951:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13952:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13953:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13954:  Function GetCurrentDirUTF8 : String
13955:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13956:  Function CreateDirUTF8( const NewDir : String) : Boolean
13957:  Function RemoveDirUTF8( const Dir : String) : Boolean
13958:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13959:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13960:  Function FileCreateUTF8( const FileName : string) : THandle;
13961:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13962:  Function ParamStrUTF8( Param : Integer) : string
13963:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13964:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
```

```
13965:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13966:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13967:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13968: end;
13969:
13970: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13971: begin
13972:   //VK_F23 = 134;
13973:   //{$EXTERNALSYM VK_F24}
13974:   //VK_F24 = 135;
13975:  TVirtualKeyCode', 'Integer
13976:  'VK_MOUSEWHEELUP','integer'(134);
13977:  'VK_MOUSEWHEELDOWN','integer'(135);
13978:  Function glIsKeyDown( c : Char) : Boolean;
13979:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13980:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13981:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13982:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13983:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13984:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13985: end;
13986:
13987: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13988: begin
13989:   TGLPoint', 'TPoint
13990:   //PGLPoint', '^TGLPoint // will not work
13991:   TGLRect', 'TRect
13992:   //PGLRect', '^TGLRect // will not work
13993:   TDelphiColor', 'TColor
13994:   TGLPicture', 'TPicture
13995:   TGLGraphic', 'TGraphic
13996:   TGLBitmap', 'TBitmap
13997:   //TGraphicClass', 'class of TGraphic
13998:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13999:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
14000:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
14001:    +'Button; Shift : TShiftState; X, Y : Integer)
14002:   TGLMouseMoveEvent', 'TMouseMoveEvent
14003:   TGLKeyEvent', 'TKeyEvent
14004:   TGLKeyPressEvent', 'TKeyPressEvent
14005:   EGLOSError', 'EWin32Error
14006:   EGLOSError', 'EWin32Error
14007:   EGLOSError', 'EOSError
14008:  'glsAllFilter','string'All // sAllFilter
14009:  Function GLPoint( const x, y : Integer) : TGLPoint
14010:  Function GLRGB( const r, g, b : Byte) : TColor
14011:  Function GLColorToRGB( color : TColor) : TColor
14012:  Function GLGetRValue( rgb : DWORD) : Byte
14013:  Function GLGetGValue( rgb : DWORD) : Byte
14014:  Function GLGetBValue( rgb : DWORD) : Byte
14015:  Procedure GLInitWinColors
14016:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
14017:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
14018:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
14019:  Procedure GLInformationDlg( const msg : String)
14020:  Function GLQuestionDlg( const msg : String) : Boolean
14021:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
14022:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
14023:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
14024:  Function GLApplicationTerminated : Boolean
14025:  Procedure GLRaiseLastOSError
14026:  Procedure GLFreeAndNil( var anObject: TObject)
14027:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
14028:  Function GLGetCurrentColorDepth : Integer
14029:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
14030:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
14031:  Procedure GLSleep( length : Cardinal)
14032:  Procedure GLQueryPerformanceCounter( var val : Int64)
14033:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
14034:  Function GLStartPrecisionTimer : Int64
14035:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
14036:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
14037:  Function GLRDTSC : Int64
14038:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
14039:  Function GLOKMessageBox( const Text, Caption : string) : Integer
14040:  Procedure GLShowHTMLUrl( Url : String)
14041:  Procedure GLShowCursor( AShow : boolean)
14042:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
14043:  Procedure GLGetCursorPos( var point : TGLPoint)
14044:  Function GLGetScreenWidth : integer
14045:  Function GLGetScreenHeight : integer
14046:  Function GLGetTickCount : int64
14047:  function RemoveSpaces(const str : String) : String;
14048:   TNormalMapSpace','( nmsObject, nmsTangent )
14049:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
14050:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
14051:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
14052:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
```

```
14053:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
14054: end;
14055:
14056: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
14057: begin
14058:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14059:  // PGLStarRecord', '^TGLStarRecord // will not work
14060:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
14061:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
14062:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
14063: end;
14064:
14065:
14066: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14067: begin
14068:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14069:   //PAABB', '^TAABB // will not work
14070:  TBSphere', 'record Center : TAffineVector; Radius : single; end
14071:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14072:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14073:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
14074:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
14075:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
14076:  Procedure SetAABB( var bb : TAABB; const v : TVector)
14077:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
14078:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
14079:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
14080:  Function BBMinX( const c : THmgBoundingBox) : Single
14081:  Function BBMaxX( const c : THmgBoundingBox) : Single
14082:  Function BBMinY( const c : THmgBoundingBox) : Single
14083:  Function BBMaxY( const c : THmgBoundingBox) : Single
14084:  Function BBMinZ( const c : THmgBoundingBox) : Single
14085:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14086:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14087:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14088:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14089:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14090:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14091:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14092:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14093:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14094:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14095:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14096:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14097:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14098:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14099:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14100:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14101:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14102:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14103:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14104:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14105:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14106:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14107:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14108:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14109:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14110:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14111:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14112:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
14113:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14114:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14115:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14116:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14117:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14118:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
       viewportSizeY:Int):TClipRect
14119: end;
14120:
14121: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14122: begin
14123:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14124:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14125:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14126:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14127:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14128:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14129:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14130:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14131:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14132:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14133:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14134:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14135:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14136:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14137:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14138:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
```

```
14139:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14140:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14141:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14142:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14143:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14144:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14145:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14146:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14147:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14148: end;
14149:
14150: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14151: begin
14152:  'EPSILON','Single').setExtended( 1e-40);
14153:  'EPSILON2','Single').setExtended( 1e-30);  }
14154:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14155:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14156:  THmgPlane', 'TVector
14157:   TDoubleHmgPlane', 'THomogeneousDblVector
14158:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14159:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14160:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14161:   TSingleArray', 'array of Single
14162:   TTransformations','array [0..15] of Single)
14163:   TPackedRotationMatrix','array [0..2] of Smallint)
14164:   TVertex', 'TAffineVector
14165:   //TVectorGL', 'THomogeneousFltVector
14166:   //TMatrixGL', 'THomogeneousFltMatrix
14167:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14168:  Function glTexPointMake( const s, t : Single) : TTexPoint
14169:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14170:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14171:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14172:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14173:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14174:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14175:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14176:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14177:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14178:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14179:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14180:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14181:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14182:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14183:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14184:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14185:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14186:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14187:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14188:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14189:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14190:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14191:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14192:  Procedure glRstVector( var v : TAffineVector);
14193:  Procedure glRstVector1( var v : TVectorGL);
14194:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14195:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14196:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14197:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14198:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14199:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14200:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14201:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14202:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14203:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14204:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14205:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14206:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Int;dest:PTexPointArray);
14207:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14208:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
      PAffineVectorArray);
14209:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14210:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14211:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14212:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14213:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14214:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14215:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14216:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14217:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14218:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14219:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14220:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14221:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14222:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14223:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14224:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
```

```
14225:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14226:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14227:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14228:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14229:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14230:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14231:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14232:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14233:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14234:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14235:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14236:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14237:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14238:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14239:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14240:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14241:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14242:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14243:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14244:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14245:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14246:  Function glLerp( const start, stop, t : Single) : Single
14247:  Function glAngleLerp( start, stop, t : Single) : Single
14248:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14249:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14250:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14251:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14252:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14253:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14254:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14255:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14256: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14257: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
       PAffineVectorArray);
14258:  Function glVectorLength( const x, y : Single) : Single;
14259:  Function glVectorLength1( const x, y, z : Single) : Single;
14260:  Function glVectorLength2( const v : TAffineVector) : Single;
14261:  Function glVectorLength3( const v : TVectorGL) : Single;
14262:  Function glVectorLength4( const v : array of Single) : Single;
14263:  Function glVectorNorm( const x, y : Single) : Single;
14264:  Function glVectorNorm1( const v : TAffineVector) : Single;
14265:  Function glVectorNorm2( const v : TVectorGL) : Single;
14266:  Function glVectorNorm3( var V : array of Single) : Single;
14267:  Procedure glNormalizeVector( var v : TAffineVector);
14268:  Procedure glNormalizeVector1( var v : TVectorGL);
14269:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14270:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14271: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14272:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14273:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14274:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14275:  Procedure glNegateVector( var V : TAffineVector);
14276:  Procedure glNegateVector2( var V : TVectorGL);
14277:  Procedure glNegateVector3( var V : array of Single);
14278:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14279:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14280:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14281:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14282:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14283:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14284:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14285:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14286:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14287:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14288:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14289:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14290:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14291:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14292:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14293:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14294:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14295:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14296:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14297:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14298:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14299:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14300:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14301:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14302:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14303:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14304:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14305:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14306:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14307:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14308:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14309:  Procedure glAbsVector( var v : TVectorGL);
14310:  Procedure glAbsVector1( var v : TAffineVector);
14311:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14312:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
```

```
14313:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14314:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14315:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14316:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14317:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14318:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14319:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14320:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14321:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14322:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14323:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14324:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14325:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14326:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14327:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14328:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14329:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14330:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14331:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14332:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14333:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14334:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14335:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14336:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14337:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14338:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14339:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14340:  Procedure glAdjointMatrix( var M : TMatrixGL);
14341:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14342:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14343:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14344:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14345:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14346:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14347:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14348:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14349:  Procedure glInvertMatrix( var M : TMatrixGL);
14350:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14351:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14352:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14353:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14354:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14355:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14356:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14357:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14358:  Procedure glNormalizePlane( var plane : THmgPlane)
14359:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14360:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14361:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14362:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14363:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14364:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14365:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14366:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14367:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14368:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14369:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14370:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14371:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14372:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14373:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14374:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14375:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14376:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14377:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14378:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14379:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14380:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14381:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14382:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14383:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14384:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14385:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14386:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14387:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14388:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14389:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14390:  Function glLnXP1( X : Extended) : Extended
14391:  Function glLog10( X : Extended) : Extended
14392:  Function glLog2( X : Extended) : Extended;
14393:  Function glLog21( X : Single) : Single;
14394:  Function glLogN( Base, X : Extended) : Extended
14395:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14396:  Function glPower( const Base, Exponent : Single) : Single;
14397:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14398:  Function glDegToRad( const Degrees : Extended) : Extended;
14399:  Function glDegToRad1( const Degrees : Single) : Single;
14400:  Function glRadToDeg( const Radians : Extended) : Extended;
```

```
14401:  Function glRadToDeg1( const Radians : Single) : Single;
14402:  Function glNormalizeAngle( angle : Single) : Single
14403:  Function glNormalizeDegAngle( angle : Single) : Single
14404:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14405:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14406:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14407:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14408:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14409:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14410:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14411:  Function glArcCos( const X : Extended) : Extended;
14412:  Function glArcCos1( const x : Single) : Single;
14413:  Function glArcSin( const X : Extended) : Extended;
14414:  Function glArcSin1( const X : Single) : Single;
14415:  Function glArcTan21( const Y, X : Extended) : Extended;
14416:  Function glArcTan21( const Y, X : Single) : Single;
14417:  Function glFastArcTan2( y, x : Single) : Single
14418:  Function glTan( const X : Extended) : Extended;
14419:  Function glTan1( const X : Single) : Single;
14420:  Function glCoTan( const X : Extended) : Extended;
14421:  Function glCoTan1( const X : Single) : Single;
14422:  Function glSinh( const x : Single) : Single;
14423:  Function glSinh1( const x : Double) : Double;
14424:  Function glCosh( const x : Single) : Single;
14425:  Function glCosh1( const x : Double) : Double;
14426:  Function glRSqrt( v : Single) : Single
14427:  Function glRLength( x, y : Single) : Single
14428:  Function glISqrt( i : Integer) : Integer
14429:  Function glILength( x, y : Integer) : Integer;
14430:  Function glILength1( x, y, z : Integer) : Integer;
14431:  Procedure glRegisterBasedExp
14432:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14433:  Function glRoundInt( v : Single) : Single;
14434:  Function glRoundInt1( v : Extended) : Extended;
14435:  Function glTrunc( v : Single) : Integer;
14436:  Function glTrunc64( v : Extended) : Int64;
14437:  Function glInt( v : Single) : Single;
14438:  Function glInt1( v : Extended) : Extended;
14439:  Function glFrac( v : Single) : Single;
14440:  Function glFrac1( v : Extended) : Extended;
14441:  Function glRound( v : Single) : Integer;
14442:  Function glRound64( v : Single) : Int64;
14443:  Function glRound641( v : Extended) : Int64;
14444:  Function glTrunc( X : Extended) : Int64
14445:  Function glRound( X : Extended) : Int64
14446:  Function glFrac( X : Extended) : Extended
14447:  Function glCeil( v : Single) : Integer;
14448:  Function glCeil64( v : Extended) : Int64;
14449:  Function glFloor( v : Single) : Integer;
14450:  Function glFloor64( v : Extended) : Int64;
14451:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14452:  Function glSign( x : Single) : Integer
14453:  Function glIsInRange( const x, a, b : Single) : Boolean;
14454:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14455:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14456:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14457:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14458:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14459:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14460:  Function glMinFloat3( const v1, v2 : Single) : Single;
14461:  Function glMinFloat4( const v : array of Single) : Single;
14462:  Function glMinFloat5( const v1, v2 : Double) : Double;
14463:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14464:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14465:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14466:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14467:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14468:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14469:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14470:  Function glMaxFloat2( const v : array of Single) : Single;
14471:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14472:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14473:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14474:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14475:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14476:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14477:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14478:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14479:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14480:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14481:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14482:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14483:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14484:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14485:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14486:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14487:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14488:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14489:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
```

```
14490:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14491:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14492:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14493:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14494:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14495:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14496:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14497:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14498:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14499:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14500:  Procedure glSortArrayAscending( var a : array of Extended)
14501:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14502:  Function glClampValue1( const aValue, aMin : Single) : Single;
14503:  Function glGeometryOptimizationMode : String
14504:  Procedure glBeginFPUOnlySection
14505:  Procedure glEndFPUOnlySection
14506:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14507:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14508:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14509:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14510:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14511:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14512:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14513:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14514:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14515:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14516:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14517:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14518:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14519:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14520:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14521:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14522:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14523:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14524:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14525:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14526:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14527:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14528:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14529:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14530:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14531:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14532:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14533:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14534:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14535:  'cPI','Single').setExtended( 3.141592654);
14536:  'cPIdiv180','Single').setExtended( 0.017453292);
14537:  'c180divPI','Single').setExtended( 57.29577951);
14538:  'c2PI','Single').setExtended( 6.283185307);
14539:  'cPIdiv2','Single').setExtended( 1.570796326);
14540:  'cPIdiv4','Single').setExtended( 0.785398163);
14541:  'c3PIdiv4','Single').setExtended( 2.35619449);
14542:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14543:  'cInv360','Single').setExtended( 1 / 360);
14544:  'c180','Single').setExtended( 180);
14545:  'c360','Single').setExtended( 360);
14546:  'cOneHalf','Single').setExtended( 0.5);
14547:  'cLn10','Single').setExtended( 2.302585093);
14548:  {'MinSingle','Extended').setExtended( 1.5e-45);
14549:  'MaxSingle','Extended').setExtended( 3.4e+38);
14550:  'MinDouble','Extended').setExtended( 5.0e-324);
14551:  'MaxDouble','Extended').setExtended( 1.7e+308);
14552:  'MinExtended','Extended').setExtended( 3.4e-4932);
14553:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14554:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14555:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14556: end;
14557:
14558: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14559: begin
14560:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14561:   (FindClass('TOBJECT'),'TFaceGroups
14562:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14563:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14564:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14565:   SIRegister_TBaseMeshObject(CL);
14566:   (FindClass('TOBJECT'),'TSkeletonFrameList
14567:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14568:   SIRegister_TSkeletonFrame(CL);
14569:   SIRegister_TSkeletonFrameList(CL);
14570:   (FindClass('TOBJECT'),'TSkeleton
14571:   (FindClass('TOBJECT'),'TSkeletonBone
14572:   SIRegister_TSkeletonBoneList(CL);
14573:   SIRegister_TSkeletonRootBoneList(CL);
```

```
14574:   SIRegister_TSkeletonBone(CL);
14575:   (FindClass('TOBJECT'),'TSkeletonColliderList
14576:   SIRegister_TSkeletonCollider(CL);
14577:   SIRegister_TSkeletonColliderList(CL);
14578:   (FindClass('TOBJECT'),'TGLBaseMesh
14579:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14580:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14581:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14582:    +'QuaternionList; end
14583:   SIRegister_TSkeleton(CL);
14584:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14585:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14586:   SIRegister_TMeshObject(CL);
14587:   SIRegister_TMeshObjectList(CL);
14588:   //TMeshObjectListClass', 'class of TMeshObjectList
14589:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14590:   SIRegister_TMeshMorphTarget(CL);
14591:   SIRegister_TMeshMorphTargetList(CL);
14592:   SIRegister_TMorphableMeshObject(CL);
14593:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14594:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14595:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14596:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14597:   SIRegister_TSkeletonMeshObject(CL);
14598:   SIRegister_TFaceGroup(CL);
14599:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14600:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14601:   SIRegister_TFGVertexIndexList(CL);
14602:   SIRegister_TFGVertexNormalTexIndexList(CL);
14603:   SIRegister_TFGIndexTexCoordList(CL);
14604:   SIRegister_TFaceGroups(CL);
14605:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14606:   SIRegister_TVectorFile(CL);
14607:   //TVectorFileClass', 'class of TVectorFile
14608:   SIRegister_TGLGLSMVectorFile(CL);
14609:   SIRegister_TGLBaseMesh(CL);
14610:   SIRegister_TGLFreeForm(CL);
14611:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14612:   TGLActorOptions', 'set of TGLActorOption
14613:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14614:   (FindClass('TOBJECT'),'TGLActor
14615:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14616:   SIRegister_TActorAnimation(CL);
14617:   TActorAnimationName', 'String
14618:   SIRegister_TActorAnimations(CL);
14619:   SIRegister_TGLBaseAnimationControler(CL);
14620:   SIRegister_TGLAnimationControler(CL);
14621:   TActorFrameInterpolation', '( afpNone, afpLinear )
14622:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14623:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14624:   SIRegister_TGLActor(CL);
14625:   SIRegister_TVectorFileFormat(CL);
14626:   SIRegister_TVectorFileFormatsList(CL);
14627:   (FindClass('TOBJECT'),'EInvalidVectorFile
14628:  Function GetVectorFileFormats : TVectorFileFormatsList
14629:  Function VectorFileFormatsFilter : String
14630:  Function VectorFileFormatsSaveFilter : String
14631:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14632:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14633:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14634: end;
14635:
14636: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14637: begin
14638:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14639:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14640:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14641:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14642:   SIRegister_TOleStream(CL);
14643:   (FindClass('TOBJECT'),'TConnectionPoints
14644:   TConnectionKind', '( ckSingle, ckMulti )
14645:   SIRegister_TConnectionPoint(CL);
14646:   SIRegister_TConnectionPoints(CL);
14647:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14648:   (FindClass('TOBJECT'),'TActiveXControlFactory
14649:   SIRegister_TActiveXControl(CL);
14650:   //TActiveXControlClass', 'class of TActiveXControl
14651:   SIRegister_TActiveXControlFactory(CL);
14652:   SIRegister_TActiveFormControl(CL);
14653:   SIRegister_TActiveForm(CL);
14654:   //TActiveFormClass', 'class of TActiveForm
14655:   SIRegister_TActiveFormFactory(CL);
14656:   (FindClass('TOBJECT'),'TPropertyPageImpl
14657:   SIRegister_TPropertyPage(CL);
14658:   //TPropertyPageClass', 'class of TPropertyPage
14659:   SIRegister_TPropertyPageImpl(CL);
14660:   SIRegister_TActiveXPropertyPage(CL);
14661:   SIRegister_TActiveXPropertyPageFactory(CL);
14662:   SIRegister_TCustomAdapter(CL);
```

```
14663:   SIRegister_TAdapterNotifier(CL);
14664:   SIRegister_IFontAccess(CL);
14665:   SIRegister_TFontAdapter(CL);
14666:   SIRegister_IPictureAccess(CL);
14667:   SIRegister_TPictureAdapter(CL);
14668:   SIRegister_TOleGraphic(CL);
14669:   SIRegister_TStringsAdapter(CL);
14670:   SIRegister_TReflectorWindow(CL);
14671:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14672:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14673:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14674:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14675:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14676:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14677:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14678:  Function ParkingWindow : HWND
14679: end;
14680:
14681: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14682: begin
14683:  // TIp6Bytes = array [0..15] of Byte;
14684: {:binary form of IPv6 adress (for string conversion routines)}
14685:  // TIp6Words = array [0..7] of Word;
14686:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14687:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14688:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14689:  Function synaIsIP( const Value : string) : Boolean');
14690:  Function synaIsIP6( const Value : string) : Boolean');
14691:  Function synaIPToID( Host : string) : Ansistring');
14692:  Function synaStrToIp6( value : string) : TIp6Bytes');
14693:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14694:  Function synaStrToIp( value : string) : integer');
14695:  Function synaIpToStr( value : integer) : string');
14696:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14697:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14698:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14699:  Function xStrToIP( const Value : String) : TIPAdr');
14700:  Function xIPToStr( const Adresse : TIPAdr) : String');
14701:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14702:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14703: end;
14704:
14705: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14706: begin
14707:  AddTypeS('TSpecials', 'set of Char');
14708:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14709:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14710:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14711:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14712:  Const('TableUU'(`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14713:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14714:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14715:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14716:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14717:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14718:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14719:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14720:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14721:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14722:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14723:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14724:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
14725:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14726:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14727:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14728:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14729:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14730:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14731:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14732:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14733:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14734:  Function synCrc32( const Value : AnsiString) : Integer');
14735:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14736:  Function Crc16( const Value : AnsiString) : Word');
14737:  Function synMD5( const Value : AnsiString) : AnsiString');
14738:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14739:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14740:  Function synSHA1( const Value : AnsiString) : AnsiString');
14741:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14742:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14743:  Function synMD4( const Value : AnsiString) : AnsiString');
14744: end;
14745:
14746: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14747: begin
14748:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14749:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14750:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14751:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
```

```
14752:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14753:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14754:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14755:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14756:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14757:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14758:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14759:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14760:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14761:    +', CP864, CP865, CP869, CP1125 )');
14762:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14763:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14764:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
       TransformTable : array of Word) : AnsiString');
14765:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
       TransformTable : array of Word; Translit : Boolean) : AnsiString');
14766:  Function GetCurCP : TMimeChar');
14767:  Function GetCurOEMCP : TMimeChar');
14768:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14769:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14770:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14771:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14772:  Function GetBOM( Value : TMimeChar) : AnsiString');
14773:  Function StringToWide( const Value : AnsiString) : WideString');
14774:  Function WideToString( const Value : WideString) : AnsiString');
14775: end;
14776:
14777: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14778: begin
14779:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14780:  Procedure WakeOnLan( MAC, IP : string)');
14781:  Function GetDNS : string');
14782:  Function GetIEProxy( protocol : string) : TProxySetting');
14783:  Function GetLocalIPs : string');
14784: end;
14785:
14786:
14787: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14788: begin
14789:  AddConstantN('synCR','Char #$0d);
14790:  Const('synLF','Char #$0a);
14791:  Const('cSerialChunk','LongInt'( 8192);
14792:  Const('LockfileDirectory','String '/var/lock');
14793:  Const('PortIsClosed','LongInt'( - 1);
14794:  Const('ErrAlreadyOwned','LongInt'( 9991);
14795:  Const('ErrAlreadyInUse','LongInt'( 9992);
14796:  Const('ErrWrongParameter','LongInt'( 9993);
14797:  Const('ErrPortNotOpen','LongInt'( 9994);
14798:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14799:  Const('ErrMaxBuffer','LongInt'( 9996);
14800:  Const('ErrTimeout','LongInt'( 9997);
14801:  Const('ErrNotRead','LongInt'( 9998);
14802:  Const('ErrFrame','LongInt'( 9999);
14803:  Const('ErrOverrun','LongInt'( 10000);
14804:  Const('ErrRxOver','LongInt'( 10001);
14805:  Const('ErrRxParity','LongInt'( 10002);
14806:  Const('ErrTxFull','LongInt'( 10003);
14807:  Const('dcb_Binary','LongWord')( $00000001);
14808:  Const('dcb_ParityCheck','LongWord')( $00000002);
14809:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14810:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14811:  Const('dcb_DtrControlMask','LongWord')( $00000030);
14812:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14813:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14814:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14815:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14816:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14817:  Const('dcb_OutX','LongWord')( $00000100);
14818:  Const('dcb_InX','LongWord')( $00000200);
14819:  Const('dcb_ErrorChar','LongWord')( $00000400);
14820:  Const('dcb_NullStrip','LongWord')( $00000800);
14821:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14822:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14823:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14824:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14825:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14826:  Const('dcb_AbortOnError','LongWord')( $00004000);
14827:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14828:  Const('synSB1','LongInt'( 0);
14829:  Const('SB1andHalf','LongInt'( 1);
14830:  Const('synSB2','LongInt'( 2);
14831:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
14832:  Const('CS7fix','LongWord')( $0000020);
14833:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14834:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14835:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14836:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14837:   //AddTypeS('PDCB', '^TDCB // will not work');
14838:  //Const('MaxRates','LongInt'( 18);
```

```
14839:  //Const('MaxRates','LongInt'( 30);
14840:  //Const('MaxRates','LongInt'( 19);
14841:  Const('O_SYNC','LongWord')( $0080);
14842:  Const('synOK','LongInt'( 0);
14843:  Const('synErr','LongInt'( integer ( - 1 ));
14844:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14845:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14846:   SIRegister_ESynaSerError(CL);
14847:   SIRegister_TBlockSerial(CL);
14848:  Function GetSerialPortNames : string');
14849: end;
14850:
14851: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14852: begin
14853:  Const('DLLIconvName','String 'libiconv.so');
14854:  Const('DLLIconvName','String 'iconv.dll');
14855:   AddTypeS('size_t', 'Cardinal');
14856:   AddTypeS('iconv_t', 'Integer');
14857:   //AddTypeS('iconv_t', 'Pointer');
14858:   AddTypeS('argptr', 'iconv_t');
14859:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14860:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14861:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14862:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14863:  Function SynaIconvClose( var cd : iconv_t) : integer');
14864:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14865:  Function IsIconvloaded : Boolean');
14866:  Function InitIconvInterface : Boolean');
14867:  Function DestroyIconvInterface : Boolean');
14868:  Const('ICONV_TRIVIALP','LongInt'( 0);
14869:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14870:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14871:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14872:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14873: end;
14874:
14875: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14876: begin
14877:  Const 'ICMP_ECHO','LongInt'( 8);
14878:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14879:  Const('ICMP_UNREACH','LongInt'( 3);
14880:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14881:  Const('ICMP6_ECHO','LongInt'( 128);
14882:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14883:  Const('ICMP6_UNREACH','LongInt'( 1);
14884:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14885:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14886:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14887:   SIRegister_TPINGSend(CL);
14888:  Function PingHost( const Host : string) : Integer');
14889:  Function TraceRouteHost( const Host : string) : string');
14890: end;
14891:
14892: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14893: begin
14894:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14895:  Const('synASN1_INT','LongWord')( $02);
14896:  Const('synASN1_OCTSTR','LongWord')( $04);
14897:  Const('synASN1_NULL','LongWord')( $05);
14898:  Const('synASN1_OBJID','LongWord')( $06);
14899:  Const('synASN1_ENUM','LongWord')( $0a);
14900:  Const('synASN1_SEQ','LongWord')( $30);
14901:  Const('synASN1_SETOF','LongWord')( $31);
14902:  Const('synASN1_IPADDR','LongWord')( $40);
14903:  Const('synASN1_COUNTER','LongWord')( $41);
14904:  Const('synASN1_GAUGE','LongWord')( $42);
14905:  Const('synASN1_TIMETICKS','LongWord')( $43);
14906:  Const('synASN1_OPAQUE','LongWord')( $44);
14907:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14908:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14909:  Function synASNEncLen( Len : Integer) : AnsiString');
14910:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14911:  Function synASNEncInt( Value : Integer) : AnsiString');
14912:  Function synASNEncUInt( Value : Integer) : AnsiString');
14913:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14914:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14915:  Function synMibToId( Mib : String) : AnsiString');
14916:  Function synIdToMib( const Id : AnsiString) : String');
14917:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14918:  Function ASNdump( const Value : AnsiString) : AnsiString');
14919:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
14920:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14921: end;
14922:
14923: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14924: begin
14925:  Const('cLDAPProtocol','String '389');
14926:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
```

```
14927:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14928:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14929:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14930:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14931:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14932:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14933:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14934:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14935:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14936:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14937:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14938:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14939:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14940:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14941:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14942:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14943:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14944:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14945:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14946:   SIRegister_TLDAPAttribute(CL);
14947:   SIRegister_TLDAPAttributeList(CL);
14948:   SIRegister_TLDAPResult(CL);
14949:   SIRegister_TLDAPResultList(CL);
14950:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14951:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14952:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14953:   SIRegister_TLDAPSend(CL);
14954:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14955: end;
14956:
14957:
14958: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14959: begin
14960:  Const('cSysLogProtocol','String '514');
14961:  Const('FCL_Kernel','LongInt'( 0);
14962:  Const('FCL_UserLevel','LongInt'( 1);
14963:  Const('FCL_MailSystem','LongInt'( 2);
14964:  Const('FCL_System','LongInt'( 3);
14965:  Const('FCL_Security','LongInt'( 4);
14966:  Const('FCL_Syslogd','LongInt'( 5);
14967:  Const('FCL_Printer','LongInt'( 6);
14968:  Const('FCL_News','LongInt'( 7);
14969:  Const('FCL_UUCP','LongInt'( 8);
14970:  Const('FCL_Clock','LongInt'( 9);
14971:  Const('FCL_Authorization','LongInt'( 10);
14972:  Const('FCL_FTP','LongInt'( 11);
14973:  Const('FCL_NTP','LongInt'( 12);
14974:  Const('FCL_LogAudit','LongInt'( 13);
14975:  Const('FCL_LogAlert','LongInt'( 14);
14976:  Const('FCL_Time','LongInt'( 15);
14977:  Const('FCL_Local0','LongInt'( 16);
14978:  Const('FCL_Local1','LongInt'( 17);
14979:  Const('FCL_Local2','LongInt'( 18);
14980:  Const('FCL_Local3','LongInt'( 19);
14981:  Const('FCL_Local4','LongInt'( 20);
14982:  Const('FCL_Local5','LongInt'( 21);
14983:  Const('FCL_Local6','LongInt'( 22);
14984:  Const('FCL_Local7','LongInt'( 23);
14985:   Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
14986:   SIRegister_TSyslogMessage(CL);
14987:   SIRegister_TSyslogSend(CL);
14988:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
      Content:string):Boolean;
14989: end;
14990:
14991:
14992: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14993: begin
14994:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14995:   SIRegister_TMessHeader(CL);
14996:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14997:   SIRegister_TMimeMess(CL);
14998: end;
14999:
15000: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
15001: begin
15002:   (FindClass('TOBJECT'),'TMimePart');
15003:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
15004:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
15005:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
15006:   SIRegister_TMimePart(CL);
15007:  Const('MaxMimeType','LongInt'( 25);
15008:  Function GenerateBoundary : string');
15009: end;
15010:
15011: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
15012: begin
15013:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
15014:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
```

```
15015:  Function NeedInline( const Value : AnsiString) : boolean');
15016:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
15017:  Function InlineCode( const Value : string) : string');
15018:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
15019:  Function InlineEmail( const Value : string) : string');
15020: end;
15021:
15022: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
15023: begin
15024:  Const('cFtpProtocol','String '21');
15025:  Const('cFtpDataProtocol','String '20');
15026:  Const('FTP_OK','LongInt'( 255);
15027:  Const('FTP_ERR','LongInt'( 254);
15028:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
15029:   SIRegister_TFTPListRec(CL);
15030:   SIRegister_TFTPList(CL);
15031:   SIRegister_TFTPSend(CL);
15032:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15033:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
15034:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
       ToPort, ToFile, ToUser, ToPass : string) : Boolean');
15035: end;
15036:
15037: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
15038: begin
15039:  Const('cHttpProtocol','String '80');
15040:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
15041:   SIRegister_THTTPSend(CL);
15042:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
15043:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
15044:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
15045:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
15046:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
       ResultData:TStrings):Bool;
15047: end;
15048:
15049: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
15050: begin
15051:  Const('cSmtpProtocol','String '25');
15052:   SIRegister_TSMTPSend(CL);
15053:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
       Passw:string):Bool;
15054:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
15055:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
       Username, Password : string):Boolean');
15056: end;
15057:
15058: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15059: begin
15060:  Const('cSnmpProtocol','String '161');
15061:  Const('cSnmpTrapProtocol','String '162');
15062:  Const('SNMP_V1','LongInt'( 0);
15063:  Const('SNMP_V2C','LongInt'( 1);
15064:  Const('SNMP_V3','LongInt'( 3);
15065:  Const('PDUGetRequest','LongWord')( $A0);
15066:  Const('PDUGetNextRequest','LongWord')( $A1);
15067:  Const('PDUGetResponse','LongWord')( $A2);
15068:  Const('PDUSetRequest','LongWord')( $A3);
15069:  Const('PDUTrap','LongWord')( $A4);
15070:  Const('PDUGetBulkRequest','LongWord')( $A5);
15071:  Const('PDUInformRequest','LongWord')( $A6);
15072:  Const('PDUTrapV2','LongWord')( $A7);
15073:  Const('PDUReport','LongWord')( $A8);
15074:  Const('ENoError',LongInt 0;
15075:  Const('ETooBig','LongInt')( 1);
15076:  Const('ENoSuchName','LongInt'( 2);
15077:  Const('EBadValue','LongInt'( 3);
15078:  Const('EReadOnly','LongInt'( 4);
15079:  Const('EGenErr','LongInt'( 5);
15080:  Const('ENoAccess','LongInt'( 6);
15081:  Const('EWrongType','LongInt'( 7);
15082:  Const('EWrongLength','LongInt'( 8);
15083:  Const('EWrongEncoding','LongInt'( 9);
15084:  Const('EWrongValue','LongInt'( 10);
15085:  Const('ENoCreation','LongInt'( 11);
15086:  Const('EInconsistentValue','LongInt'( 12);
15087:  Const('EResourceUnavailable','LongInt'( 13);
15088:  Const('ECommitFailed','LongInt'( 14);
15089:  Const('EUndoFailed','LongInt'( 15);
15090:  Const('EAuthorizationError','LongInt'( 16);
15091:  Const('ENotWritable','LongInt'( 17);
15092:  Const('EInconsistentName','LongInt'( 18);
15093:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15094:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15095:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15096:   SIRegister_TSNMPMib(CL);
15097:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15098:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15099:   SIRegister_TSNMPRec(CL);
```

```
15100:   SIRegister_TSNMPSend(CL);
15101:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15102:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15103:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15104:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15105:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15106:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15107:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
       Integer; const MIBName, MIBValue : TStringList) : Integer');
15108: end;
15109:
15110: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15111: begin
15112:  Function GetDomainName2: AnsiString');
15113:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15114:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15115:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15116:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15117:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15118: end;
15119:
15120: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15121: begin
15122:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15123:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15124:  Function wwStrToDate( const S : string) : boolean');
15125:  Function wwStrToTime( const S : string) : boolean');
15126:  Function wwStrToDateTime( const S : string) : boolean');
15127:  Function wwStrToTimeVal( const S : string) : TDateTime');
15128:  Function wwStrToDateVal( const S : string) : TDateTime');
15129:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15130:  Function wwStrToInt( const S : string) : boolean');
15131:  Function wwStrToFloat( const S : string) : boolean');
15132:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15133:  Function wwNextDay( Year, Month, Day : Word) : integer');
15134:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15135:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15136:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15137:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15138:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15139:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15140:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15141:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15142:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15143: end;
15144:
15145: unit uPSI_Themes;
15146: Function ThemeServices : TThemeServices');
15147: Function ThemeControl( AControl : TControl) : Boolean');
15148: Function UnthemedDesigner( AControl : TControl) : Boolean');
15149: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15150: begin
15151:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15152: end;
15153: Unit uPSC_menus;
15154:  Function StripHotkey( const Text : string) : string');
15155:  Function GetHotkey( const Text : string) : string');
15156:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15157:  Function IsAltGRPressed : boolean');
15158:
15159: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15160: begin
15161:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15162:   SIRegister_TIdIMAP4Server(CL);
15163: end;
15164:
15165: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15166: begin
15167:  'HASH_SIZE','LongInt'( 256);
15168:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15169:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15170:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15171:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15172:    +' : Integer; Value : Variant; end');
15173:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15174:   SIRegister_TVariantSymbolTable(CL);
15175: end;
15176:
15177: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15178: begin
15179:   SIRegister_TThreadLocalVariables(CL);
15180:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15181:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15182:  Function ThreadLocals : TThreadLocalVariables');
15183:  Procedure WriteDebug( sz : String)');
15184:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15185:  'UDF_FAILURE','LongInt'( 1);
15186:  'cSignificantlyLarger','LongInt'( 1024 * 4);
```

```
15187:  CL.AddTypeS('mTByteArray', 'array of byte;');
15188:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15189:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15190:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15191:  function IsNetworkConnected: Boolean;
15192:  function IsInternetConnected: Boolean;
15193:  function IsCOMConnected: Boolean;
15194:  function IsNetworkOn: Boolean;
15195:  function IsInternetOn: Boolean;
15196:  function IsCOMOn: Boolean;
15197:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15198:  TmrProc', 'procedure TmrProc(hWnd: HWND; uMsg: Integer; idEvent: Integer; dwTime: Integer);');
15199:  Function SetTimer2( hWnd : HWND; nIDEvent, uElapse : UINT; lpTimerFunc : TmrProc) : UINT');
15200:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15201:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15202:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15203:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15204:  Function GetMenu( hWnd : HWND) : HMENU');
15205:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15206: end;
15207:
15208: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15209: begin
15210:   SIRegister_IDataBlock(CL);
15211:   SIRegister_ISendDataBlock(CL);
15212:   SIRegister_ITransport(CL);
15213:   SIRegister_TDataBlock(CL);
15214:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15215:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15216:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15217:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15218:   SIRegister_TCustomDataBlockInterpreter(CL);
15219:   SIRegister_TSendDataBlock(CL);
15220:   'CallSig','LongWord')( $D800);
15221:   'ResultSig','LongWord')( $D400);
15222:   'asMask','LongWord')( $00FF);
15223:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15224:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15225:  Procedure CheckSignature( Sig : Integer)');
15226: end;
15227:
15228: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15229: begin
15230:   //CL.AddTypeS('HINTERNET', '___Pointer');
15231:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15232:   CL.AddTypeS('HINTERNET', 'Integer');
15233:   CL.AddTypeS('HINTERNET2', '___Pointer');
15234:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15235:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15236:   CL.AddTypeS('INTERNET_PORT', 'Word');
15237:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15238:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15239:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15240:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15241:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15242:  Function InternetCrackUrl(lpszUrl:PChar;dwUrlLength,dwFlags:DWORD;var
       lpUrlComponents:TURLComponents):BOOL;
15243:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
       dwUrlLength:DWORD):BOOL;
15244:  Function InternetCloseHandle(hInet : HINTERNET) : BOOL');
15245:  Function
       InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
       lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15246:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15247:  ;dwContext:DWORD):HINTERNET;
15248:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
       lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15249:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
       dwContext:DWORD):BOOL;
15250:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15251:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15252:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15253:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15254:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15255:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15256:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15257:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
       lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15258:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
       lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15259:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15260:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15261:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
       dwContext:DWORD):BOOL;
15262:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
       TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15263:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
       BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
```

```
15264:  Function
        WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15265:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15266:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15267:  Function
        FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15268:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15269:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15270:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15271:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15272:  Function
        FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15273:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15274:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15275:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15276:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15277:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15278:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15279:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15280:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15281:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15282:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15283:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15284:  Function
        GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
        PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15285:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15286:  Function
        GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15287:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
        PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15288:  Function
        HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15289:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
        dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15290:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15291:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15292:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15293:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15294:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15295:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15296:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15297:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
        lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15298:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
        lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15299:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15300:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15301:  Function
        InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15302:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15303: end;
15304:
15305: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15306: begin
15307:   AddTypeS('strCharSet', 'set of char');
15308:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15309:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15310:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15311:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15312:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15313:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15314:  Procedure strStripWhiteSpace( var s : string)');
15315:  Function strRemoveChar( str : string; removeChar : char) : string');
15316:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15317:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15318:  Function wwEqualStr( s1, s2 : string) : boolean');
15319:  Function strCount( s : string; delimeter : char) : integer');
15320:  Function strWhiteSpace : strCharSet');
15321:  Function wwExtractFileNameOnly( const FileName : string) : string');
15322:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15323:  Function strTrailing( s : string; delimeter : char) : string');
15324:  Function strPreceding( s : string; delimeter : char) : string');
15325:  Function wwstrReplace( s, Find, Replace : string) : string');
15326: end;
15327:
15328: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15329: begin
15330:   SIRegister_TRemoteDataModule(CL);
15331:   SIRegister_TCRemoteDataModule(CL);
15332:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15333:  Procedure UnregisterPooled( const ClassID : string)');
15334:  Procedure EnableSocketTransport( const ClassID : string)');
15335:  Procedure DisableSocketTransport( const ClassID : string)');
15336:  Procedure EnableWebTransport( const ClassID : string)');
15337:  Procedure DisableWebTransport( const ClassID : string)');
15338: end;
15339:
15340: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
```

```
15341: begin
15342:  Function mxArcCos( x : Real) : Real');
15343:  Function mxArcSin( x : Real) : Real');
15344:  Function Comp2Str( N : Comp) : String');
15345:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15346:  Function Int2Str( N : LongInt) : String');
15347:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15348:  Function LogXY( x, y : Real) : Real');
15349:  Function Pennies2Dollars( C : Comp) : String');
15350:  Function mxPower( X : Integer; Y : Integer) : Real');
15351:  Function Real2Str( N : Real; Width, Places : integer) : String');
15352:  Function mxStr2Comp( MyString : string) : Comp');
15353:  Function mxStr2Pennies( S : String) : Comp');
15354:  Function Str2Real( MyString : string) : Real');
15355:  Function XToTheY( x, y : Real) : Real');
15356: end;
15357:
15358: //***********************Cindy Functions!*****************************
15359: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15360: begin
15361:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15362:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15363:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15364:  MessagePlainText','String 'text/plain');
15365:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15366:  MessageAlterText_Html','String 'multipart/alternative');
15367:  MessageHtml_Attach','String 'multipart/mixed');
15368:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15369:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15370:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15371:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15372:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15373:  Function ForceDecodeHeader( aHeader : String) : String');
15374:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15375:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15376:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15377:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15378:  Function Get_MD5( const aFileName : string) : string');
15379:  Function Get_MD5FromString( const aString : string) : string');
15380: end;
15381:
15382: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15383: begin
15384:  Function IsFolder( SRec : TSearchrec) : Boolean');
15385:  Function isFolderReadOnly( Directory : String) : Boolean');
15386:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15387:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15388:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15389:  Function DiskFreeBytes( Drv : Char) : Int64');
15390:  Function DiskBytes( Drv : Char) : Int64');
15391:  Function GetFileBytes( Filename : String) : Int64');
15392:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15393:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15394:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15395:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15396:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15397:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15398:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15399:  SE_TCB_NAME','String 'SeTcbPrivilege');
15400:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15401:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15402:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15403:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15404:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15405:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15406:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15407:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15408:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15409:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15410:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15411:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15412:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
15413:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15414:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15415:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15416:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15417:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15418:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15419:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15420:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15421: end;
15422:
15423:
15424: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15425: begin
15426:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15427:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15428:  Function ShellGetExtensionName( FileName : String) : String');
15429:  Function ShellGetIconIndex( FileName : String) : Integer');
```

```
15430:  Function ShellGetIconHandle( FileName : String) : HIcon');
15431:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15432:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15433:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15434:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15435:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15436:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15437:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15438:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
       WaitCloseCompletion : boolean) : Boolean');
15439:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15440:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15441:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15442:  Function GetModificationDate( Filename : String) : TDateTime');
15443:  Function GetCreationDate( Filename : String) : TDateTime');
15444:  Function GetLastAccessDate( Filename : String) : TDateTime');
15445:  Function FileDelete( Filename : String) : Boolean');
15446:  Function FileIsOpen( Filename : string) : boolean');
15447:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15448:  Function DirectoryDelete( Directory : String) : Boolean');
15449:  Function GetPrinters( PrintersList : TStrings) : Integer');
15450:  Procedure SetDefaultPrinter( PrinterName : String)');
15451:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15452:  Function WinToDosPath( WinPathName : String) : String');
15453:  Function DosToWinPath( DosPathName : String) : String');
15454:  Function cyGetWindowsVersion : TWindowsVersion');
15455:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15456:  Procedure WindowsShutDown( Restart : boolean)');
15457:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
       FileIcon:string;NumIcone:integer);
15458:  Procedure GetWindowsFonts( FontsList : TStrings)');
15459:  Function GetAvailableFilename( DesiredFileName : String): String');
15460: end;
15461:
15462: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15463: begin
15464:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15465:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15466:   Type(TStringRead', '( srFromLeft, srFromRight )');
15467:   Type(TStringReads', 'set of TStringRead');
15468:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15469:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15470:   Type(TWordsOptions', 'set of TWordsOption');
15471:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15472:   Type(TCarTypes', 'set of TCarType');
15473:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15474:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15475:  Function Char_GetType( aChar : Char) : TCarType');
15476:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15477:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15478:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15479:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15480:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15481:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15482:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15483:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15484:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15485:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15486:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15487:  Function String_Quote( Str : String) : String');
15488:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15489:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15490:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15491:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15492:  Function String_ToInt( Str : String) : Integer');
15493:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15494:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15495:  Function String_Reverse( Str : String) : String');
15496:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15497:  Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15498:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15499:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15500:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
       Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15501:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15502:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15503:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15504:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15505:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15506:  Function String_End( Str : String; Cars : Word) : String');
15507:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15508:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15509:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
15510:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15511:  Function String_IsNumbers( Str : String) : Boolean');
15512:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15513:  Function StringToCsvCell( aStr : String) : String');
```

```
15514: end;
15515:
15516: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15517: begin
15518:  Function LongDayName( aDate : TDate) : String');
15519:  Function LongMonthName( aDate : TDate) : String');
15520:  Function ShortYearOf( aDate : TDate) : byte');
15521:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15522:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15523:  Function SecondsToMinutes( Seconds : Integer) : Double');
15524:  Function MinutesToSeconds( Minutes : Double) : Integer');
15525:  Function MinutesToHours( Minutes : Integer) : Double');
15526:  Function HoursToMinutes( Hours : Double) : Integer');
15527:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15528:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15529:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15530:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15531:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15532:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15533:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15534:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15535:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15536: end;
15537:
15538: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15539: begin
15540:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15541:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15542:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15543:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15544:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15545:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15546:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15547:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15548:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15549:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15550:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15551:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15552:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15553:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
15554:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15555:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15556:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15557:  Procedure cyCenterControl( aControl : TControl)');
15558:  Function GetLastParent( aControl : TControl) : TWinControl');
15559:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15560:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15561: end;
15562:
15563: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15564: begin
15565:  Function TablePackTable( Tab : TTable) : Boolean');
15566:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15567:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15568:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15569:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15570:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15571:  Function TableEmptyTable( Tab : TTable) : Boolean');
15572:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15573:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15574:  Function
       TableCreate(Owner:TComponent;DBaseName:ShortString;TblName:String;IdxName:ShortString;ReadOnly:Boolean):TTable;
15575:  Function
       TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15576:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15577: end;
15578:
15579: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15580: begin
15581:   SIRegister_TcyRunTimeDesign(CL);
15582:   SIRegister_TcyShadowText(CL);
15583:   SIRegister_TcyBgPicture(CL);
15584:   SIRegister_TcyGradient(CL);
15585:   SIRegister_tcyBevel(CL);
15586:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15587:   SIRegister_tcyBevels(CL);
15588:   SIRegister_TcyImagelistOptions(CL);
15589:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15590: end;
15591:
15592: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15593: begin
15594:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
       adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
       Maxdegrade : Byte;'+
15595:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
```

```
15596:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15597:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15598:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
        Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15599:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
        Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15600:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15601:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15602:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
        BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15603:  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;
15604:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
        aState : TButtonState; Focused, Hot : Boolean)');
15605:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
        aState : TButtonState; Focused, Hot : Boolean)');
15606:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
        TColor; aState : TButtonState; Focused, Hot : Boolean)');
15607:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
        GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15608:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
        const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15609:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
        TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15610:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
        TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15611:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
        WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15612:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15613:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15614:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15615:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
        CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15616:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15617:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
        Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15618:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15619:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15620:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15621:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15622:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15623:  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);
15624:  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;
15625:  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);');
15626:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15627:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15628:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15629:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15630:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15631:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15632:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15633:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15634:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15635:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15636:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15637:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15638:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15639:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15640:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15641:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15642: end;
15643:
15644: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15645: begin
15646:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15647:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15648:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15649:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15650:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15651:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15652:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15653:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15654:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15655:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15656:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15657:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
        bmInvertReverseFromColor);
15658:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
        rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
```

```
15659:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15660:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15661: end;
15662:
15663: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15664: begin
15665:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15666:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15667:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15668:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15669:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15670:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15671:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15672:  Const SC_GROUP_IDENTIFIERW','String') '+');
15673:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15674:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15675:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15676:  Const SERVICE_INACTIVE','LongWord $00000002);
15677:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15678:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15679:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15680:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15681:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15682:  Const SERVICE_STOPPED','LongWord $00000001);
15683:  Const SERVICE_START_PENDING','LongWord $00000002);
15684:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15685:  Const SERVICE_RUNNING','LongWord $00000004);
15686:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15687:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15688:  Const SERVICE_PAUSED','LongWord $00000007);
15689:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15690:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15691:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15692:  Const SC_MANAGER_CONNECT','LongWord $0001);
15693:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15694:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15695:  Const SC_MANAGER_LOCK','LongWord $0008);
15696:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15697:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15698:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15699:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15700:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15701:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15702:  Const SERVICE_START','LongWord $0010);
15703:  Const SERVICE_STOP','LongWord $0020);
15704:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15705:  Const SERVICE_INTERROGATE','LongWord $0080);
15706:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15707:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15708:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15709:  Const SERVICE_ADAPTER','LongWord $00000004);
15710:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15711:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15712:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15713:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15714:  Const SERVICE_BOOT_START','LongWord $00000000);
15715:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15716:  Const SERVICE_AUTO_START','LongWord $00000002);
15717:  Const SERVICE_DEMAND_START','LongWord $00000003);
15718:  Const SERVICE_DISABLED','LongWord $00000004);
15719:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15720:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15721:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15722:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15723:   CL.AddTypeS('SC_HANDLE', 'THandle');
15724:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15725:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15726:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15727:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15728:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15729:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15730:   Const TServiceStatus', '_SERVICE_STATUS');
15731:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15732:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15733:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15734:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15735:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15736:   TEnumServiceStatus', 'TEnumServiceStatusA');
15737:   SC_LOCK', '___Pointer');
15738:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15739:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15740:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15741:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15742:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15743:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15744:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15745:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15746:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15747:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
```

```
15748:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15749:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15750:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15751:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15752:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15753:   TQueryServiceConfig', 'TQueryServiceConfigA');
15754:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15755:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15756:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
       dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15757:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15758:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
       dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15759:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15760:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15761:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
       TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15762:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
       lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
       : DWORD):BOOL');
15763:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
       lpcchBuffer:DWORD):BOOL');
15764:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
       lpcchBuffer:DWORD):BOOL;
15765:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15766:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15767:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15768:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15769:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
       cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15770:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15771:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15772:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15773:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15774: end;
15775:
15776: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15777: begin
15778:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
       AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
       BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15779:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
       DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15780:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15781:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
       TWinControl;
15782:  Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
       AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15783:  Function CreateNotifyThread(const
       FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15784: end;
15785:
15786: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15787: begin
15788:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15789:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15790:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15791:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15792:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15793:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15794:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15795:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15796:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15797:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15798:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15799:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15800:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15801:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15802:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15803:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15804:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15805:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15806:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15807:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15808:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15809:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15810:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15811:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15812:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15813:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15814:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15815:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15816:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
15817:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15818:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15819:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15820:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15821:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15822:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
```

```
15823:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15824:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15825:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15826:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15827:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15828:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15829:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15830:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15831:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15832:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15833:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
       List:TStrings):Bool
15834:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15835:   FindClass('TOBJECT'),'EJclFileSummaryError');
15836:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15837:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15838:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15839:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15840:   SIRegister_TJclFilePropertySet(CL);
15841:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15842:   SIRegister_TJclFileSummary(CL);
15843:   SIRegister_TJclFileSummaryInformation(CL);
15844:   SIRegister_TJclDocSummaryInformation(CL);
15845:   SIRegister_TJclMediaFileSummaryInformation(CL);
15846:   SIRegister_TJclMSISummaryInformation(CL);
15847:   SIRegister_TJclShellSummaryInformation(CL);
15848:   SIRegister_TJclStorageSummaryInformation(CL);
15849:   SIRegister_TJclImageSummaryInformation(CL);
15850:   SIRegister_TJclDisplacedSummaryInformation(CL);
15851:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15852:   SIRegister_TJclMiscSummaryInformation(CL);
15853:   SIRegister_TJclWebViewSummaryInformation(CL);
15854:   SIRegister_TJclMusicSummaryInformation(CL);
15855:   SIRegister_TJclDRMSummaryInformation(CL);
15856:   SIRegister_TJclVideoSummaryInformation(CL);
15857:   SIRegister_TJclAudioSummaryInformation(CL);
15858:   SIRegister_TJclControlPanelSummaryInformation(CL);
15859:   SIRegister_TJclVolumeSummaryInformation(CL);
15860:   SIRegister_TJclShareSummaryInformation(CL);
15861:   SIRegister_TJclLinkSummaryInformation(CL);
15862:   SIRegister_TJclQuerySummaryInformation(CL);
15863:   SIRegister_TJclImageInformation(CL);
15864:   SIRegister_TJclJpegSummaryInformation(CL);
15865: end;
15866:
15867: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15868: begin
15869:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15870:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15871:   T8087Infinity','( icProjective, icAffine )');
15872:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15873:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15874:  Function Get8087ControlWord : Word');
15875:  Function Get8087Infinity : T8087Infinity');
15876:  Function Get8087Precision : T8087Precision');
15877:  Function Get8087Rounding : T8087Rounding');
15878:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15879:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15880:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15881:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15882:  Function Set8087ControlWord( const Control : Word) : Word');
15883:  Function ClearPending8087Exceptions : T8087Exceptions');
15884:  Function GetPending8087Exceptions : T8087Exceptions');
15885:  Function GetMasked8087Exceptions : T8087Exceptions');
15886:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15887:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15888:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15889: end;
15890:
15891: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15892: begin
15893:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15894:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15895:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
       Accept:Bool;Sorted:Bool;
15896:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15897:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15898:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15899:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15900:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15901: end;
15902:
15903: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15904: begin
15905:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15906:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15907:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15908:   type ULONG', 'Cardinal');
15909:      LPCWSTR', 'PChar');
```

```
15910:   CL.AddTypeS('LPWSTR', 'PChar');
15911:   LPSTR', 'PChar');
15912:   TBindVerb', 'ULONG');
15913:   TBindInfoF', 'ULONG');
15914:   TBindF', 'ULONG');
15915:   TBSCF', 'ULONG');
15916:   TBindStatus', 'ULONG');
15917:   TCIPStatus', 'ULONG');
15918:   TBindString', 'ULONG');
15919:   TPiFlags', 'ULONG');
15920:   TOIBdgFlags', 'ULONG');
15921:   TParseAction', 'ULONG');
15922:   TPSUAction', 'ULONG');
15923:   TQueryOption', 'ULONG');
15924:   TPUAF', 'ULONG');
15925:   TSZMFlags', 'ULONG');
15926:   TUrlZone', 'ULONG');
15927:   TUrlTemplate', 'ULONG');
15928:   TZAFlags', 'ULONG');
15929:   TUrlZoneReg', 'ULONG');
15930:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15931:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15932:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15933:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15934:  const 'CF_NULL','LongInt').SetInt( 0);
15935:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15936:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15937:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15938:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15939:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15940:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15941:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15942:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15943:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15944:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15945:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15946:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15947:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15948:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15949:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15950:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15951:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15952:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15953:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15954:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15955:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15956:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15957:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15958:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15959:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15960:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15961:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15962:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15963:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15964:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15965:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15966:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15967:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15968:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15969:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15970:   SIRegister_IPersistMoniker(CL);
15971:   SIRegister_IBindProtocol(CL);
15972:   SIRegister_IBinding(CL);
15973:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15974:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15975:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15976:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15977:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15978:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15979:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15980:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15981:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
15982:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15983:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15984:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15985:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
15986:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
15987:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
15988:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
15989:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
15990:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
15991:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
15992:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
15993:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
15994:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
15995:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
15996:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
15997:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
15998:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
```

```
15999:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
16000:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
16001:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
16002:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
16003:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
16004:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
16005:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
16006:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
16007:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
16008:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
16009:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
16010:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
16011:  const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
16012:  const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
16013:  const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
16014:  const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
16015:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
16016:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
16017:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
16018:  const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
16019:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
16020:  const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
16021:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
16022:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
16023:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
16024:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
16025:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
16026:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
16027:  const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
16028:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
16029:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
16030:  const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
16031:  const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
16032:  const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
16033:  const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
16034:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
16035:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
16036:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
16037:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
16038:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
16039:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
16040:  const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
16041:  const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
16042:  const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
16043:  const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
16044:  const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
16045:  const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
16046:  const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
16047:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
16048:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
16049:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
16050:  const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
16051:  const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
16052:  const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
16053:  const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
16054:  const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
16055:  const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
16056:  const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
16057:  // PBindInfo', '^TBindInfo // will not work');
16058:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
16059:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16060:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16061:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16062:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16063:   TBindInfo', '_tagBINDINFO');
16064:   BINDINFO', '_tagBINDINFO');}
16065:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16066:    +'criptor : DWORD; bInheritHandle : BOOL; end');
16067:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
16068:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
16069:   //PRemBindInfo', '^TRemBindInfo // will not work');
16070:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16071:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16072:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16073:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
16074:    +'n; dwReserved : DWORD; end');
16075:   TRemBindInfo', '_tagRemBINDINFO');
16076:   RemBINDINFO', '_tagRemBINDINFO');}
16077:   //PRemFormatEtc', '^TRemFormatEtc // will not work');
16078:   tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
16079:   TRemFormatEtc', 'tagRemFORMATETC');
16080:   RemFORMATETC', 'tagRemFORMATETC');
16081:   SIRegister_IBindStatusCallback(CL);
16082:   SIRegister_IAuthenticate(CL);
16083:   SIRegister_IHttpNegotiate(CL);
16084:   SIRegister_IWindowForBindingUI(CL);
16085:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16086:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16087:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
```

```
16088:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16089:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16090:  const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16091:  const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT',LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16092:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16093:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16094:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16095:   SIRegister_ICodeInstall(CL);
16096:   SIRegister_IWinInetInfo(CL);
16097:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16098:   SIRegister_IHttpSecurity(CL);
16099:   SIRegister_IWinInetHttpInfo(CL);
16100:   SIRegister_IBindHost(CL);
16101:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16102:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16103:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16104:  Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback) : HResult');
16105:  Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback) : HResult');
16106:  Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
       : IBindStatusCallback) : HResult');
16107:  Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
       IBindStatusCallback) : HResult');
16108:  Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
       p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16109:  Function HlinkGoBack( unk : IUnknown) : HResult');
16110:  Function HlinkGoForward( unk : IUnknown) : HResult');
16111:  Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult');
16112: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult');
16113:   SIRegister_IInternet(CL);
16114:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16115:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16116:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16117:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16118:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16119:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16120:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16121:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16122:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16123:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16124:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16125:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16126:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16127:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16128:   //POLEStrArray', '^TOLESTRArray // will not work');
16129:   SIRegister_IInternetBindInfo(CL);
16130:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16131:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16132:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16133:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16134:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16135:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16136:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16137:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16138:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16139:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16140:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16141:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16142:   //PProtocolData', '^TProtocolData // will not work');
16143:   _tagPROTOCOLDATA','record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16144:   TProtocolData', '_tagPROTOCOLDATA');
16145:   PROTOCOLDATA', '_tagPROTOCOLDATA');
16146:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink, 'IInternetProtocolSink');
16147:   SIRegister_IInternetProtocolRoot(CL);
16148:   SIRegister_IInternetProtocol(CL);
16149:   SIRegister_IInternetProtocolSink(CL);
16150:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16151:   SIRegister_IInternetSession(CL);
16152:   SIRegister_IInternetThreadSwitch(CL);
16153:   SIRegister_IInternetPriority(CL);
16154:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16155:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16156:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16157:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16158:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16159:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16160:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16161:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16162:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16163:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16164:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16165:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16166:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16167:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16168:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16169:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16170:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16171:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16172:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16173:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
```

```
16174:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16175:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16176:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16177:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16178:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16179:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16180:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16181:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16182:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16183:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16184:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16185:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16186:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16187:   SIRegister_IInternetProtocolInfo(CL);
16188:   IOInet', 'IInternet');
16189:   IOInetBindInfo', 'IInternetBindInfo');
16190:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16191:   IOInetProtocol', 'IInternetProtocol');
16192:   IOInetProtocolSink', 'IInternetProtocolSink');
16193:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16194:   IOInetSession', 'IInternetSession');
16195:   IOInetPriority', 'IInternetPriority');
16196:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16197:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16198:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16199:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16200:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16201:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16202:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16203:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16204:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16205:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16206:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16207:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16208:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16209:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16210:   //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16211:  //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16212: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16213: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16214:  //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16215:  //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16216:  //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16217:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16218:   SIRegister_IInternetSecurityMgrSite(CL);
16219:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16220:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16221:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16222:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16223:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16224:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16225:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16226:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16227:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16228:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16229:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16230:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16231:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16232:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16233:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16234:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16235:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16236:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16237:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16238:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16239:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16240:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16241:  const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16242:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16243:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16244:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16245:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16246:   SIRegister_IInternetSecurityManager(CL);
16247:   SIRegister_IInternetHostSecurityManager(CL);
16248:   SIRegister_IInternetSecurityManagerEx(CL);
16249:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16250:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16251:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16252:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16253:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
```

```
16254:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16255:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16256:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16257:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16258:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16259:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16260:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16261:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16262:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16263:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16264:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16265:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16266:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16267:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16268:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16269:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16270:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16271:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16272:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16273:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16274:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16275:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16276:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16277:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16278:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16279:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16280:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16281:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16282:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16283:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16284:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16285:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16286:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16287:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16288:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16289:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16290:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16291:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16292:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16293:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16294:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16295:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16296:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16297:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16298:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16299:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16300:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16301:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16302:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16303:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16304:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16305:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16306:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16307:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16308:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16309:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16310:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16311:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16312:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16313:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16314:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16315:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16316:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16317:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16318:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16319:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16320:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16321:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16322:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16323:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16324:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16325:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16326:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16327:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16328:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16329:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16330:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16331:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16332:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16333:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16334:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16335:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16336:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16337:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16338:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16339:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16340:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16341:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16342:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
```

```
16343:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16344:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16345:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16346:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16347:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16348:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16349:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16350:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16351:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16352:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16353:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16354:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16355:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16356:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16357:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16358:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16359:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16360:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16361:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16362:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16363:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16364:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16365:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16366:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16367:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16368:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16369:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16370:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16371:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16372:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16373:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16374:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16375:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16376:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16377:   //PZoneAttributes', '^TZoneAttributes // will not work');
16378:  _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');
16379:   { _ZONEATTRIBUTES = packed record
16380:     cbSize: ULONG;
16381:     szDisplayName: array [0..260 - 1] of WideChar;
16382:     szDescription: array [0..200 - 1] of WideChar;
16383:     szIconPath: array [0..260 - 1] of WideChar;
16384:     dwTemplateMinLevel: DWORD;
16385:     dwTemplateRecommended: DWORD;
16386:     dwTemplateCurrentLevel: DWORD;
16387:     dwFlags: DWORD;
16388:   end; }
16389:   TZoneAttributes', '_ZONEATTRIBUTES');
16390:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16391:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16392:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16393:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16394:   SIRegister_IInternetZoneManager(CL);
16395:   SIRegister_IInternetZoneManagerEx(CL);
16396:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16397:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16398:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16399:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16400:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16401:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16402:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16403:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16404:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16405:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16406:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16407:   TCodeBaseHold', '_tagCODEBASEHOLD');
16408:   CODEBASEHOLD', '_tagCODEBASEHOLD');
16409:   //PSoftDistInfo', '^TSoftDistInfo // will not work');
16410:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16411:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16412:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
16413:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16414:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16415:   TSoftDistInfo', '_tagSOFTDISTINFO');
16416:   SOFTDISTINFO', '_tagSOFTDISTINFO');
16417:   SIRegister_ISoftDistExt(CL);
16418:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16419:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
       dwAdvertisedVersionLS : DWORD) : HResult');
16420:   SIRegister_IDataFilter(CL);
16421:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16422:   _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
16423:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16424:    +'terFlags : DWORD; end');
16425:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16426:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16427:   //PDataInfo', '^TDataInfo // will not work');
16428:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
```

```
16429:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16430:   TDataInfo', '_tagDATAINFO');
16431:   DATAINFO', '_tagDATAINFO');
16432:   SIRegister_IEncodingFilterFactory(CL);
16433:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16434:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16435:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16436:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16437:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16438:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16439:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16440:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16441:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16442: end;
16443:
16444: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16445: begin
16446:  Procedure reformatMemo( const m : TCustomMemo)');
16447:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16448:  Procedure MoveToTop( memo : TMemo)');
16449:  Procedure ScrollToTop( memo : TMemo)');
16450:  Function LineNumberClicked( memo : TMemo) : integer');
16451:  Function MemoClickedLine( memo : TMemo) : integer');
16452:  Function ClickedMemoLine( memo : TMemo) : integer');
16453:  Function MemoLineClicked( memo : TMemo) : integer');
16454:  Function LinePositionClicked( Memo : TMemo) : integer');
16455:  Function ClickedMemoPosition( memo : TMemo) : integer');
16456:  Function MemoPositionClicked( memo : TMemo) : integer');
16457:  Procedure AdjustGridSize( grid : TDrawGrid)');
16458:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16459:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16460:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16461:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16462:  Procedure sortstrDown( var s : string)');
16463:  Procedure sortstrUp( var s : string)');
16464:  Procedure rotatestrleft( var s : string)');
16465:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16466:  Function deblank( s : string) : string');
16467:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16468:  Procedure FreeAndClearListBox( C : TListBox);');
16469:  Procedure FreeAndClearMemo( C : TMemo);');
16470:  Procedure FreeAndClearStringList( C : TStringList);');
16471:  Function dffgetfilesize( f : TSearchrec) : int64');
16472: end;
16473:
16474: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16475: begin
16476:   CL.AddTypeS('intset', 'set of byte');
16477:   TPoint64', 'record x : int64; y : int64; end');
16478:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16479:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16480:  Function GeneratePentagon( n : integer) : integer');
16481:  Function IsPentagon( p : integer) : boolean');
16482:  Function isSquare( const N : int64) : boolean');
16483:  Function isCube( const N : int64) : boolean');
16484:  Function isPalindrome( const n : int64) : boolean;');
16485:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16486:  Function GetEulerPhi( n : int64) : int64');
16487:  Function dffIntPower( a, b : int64) : int64;');
16488:  Function IntPower1( a : extended; b : int64) : extended;');
16489:  Function gcd2( a, b : int64) : int64');
16490:  Function GCDMany( A : array of integer) : integer');
16491:  Function LCMMany( A : array of integer) : integer');
16492:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16493:  Function dffFactorial( n : int64) : int64');
16494:  Function digitcount( n : int64) : integer');
16495:  Function nextpermute( var a : array of integer) : boolean');
16496:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16497:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16498:  Function InttoBinaryStr( nn : integer) : string');
16499:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16500:  Function AngleToStr( angle : extended) : string');
16501:  Function deg2rad( deg : extended) : extended');
16502:  Function rad2deg( rad : extended) : extended');
16503:  Function GetLongToMercProjection( const long : extended) : extended');
16504:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16505:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16506:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16507:   SIRegister_TPrimes(CL);
16508:   //RIRegister_TPrimes(CL);
16509:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16510:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16511:  Function Random64( const N : Int64) : Int64;');
16512:  Procedure Randomize64');
16513:  Function Random641 : extended;');
16514:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16515: end;
16516:
16517: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
```

```
16518: begin
16519:   TrealPoint', 'record x : extended; y : extended; end');
16520:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16521:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16522:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16523:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16524:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16525:  Function realpoint( x, y : extended) : TRealPoint');
16526:  Function dist( const p1, p2 : TrealPoint) : extended');
16527:  Function intdist( const p1, p2 : TPoint) : integer');
16528:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16529:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16530:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16531:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16532:  Function GetTheta( const L : TLine) : extended;');
16533:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16534:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16535:  Procedure Extendline( var L : TLine; dist : integer);');
16536:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16537:  Function Linesintersect( line1, line2 : TLine) : boolean');
16538:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16539:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16540:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16541:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16542:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16543:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16544:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16545:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16546:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16547:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16548:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16549:  Function DegtoRad( d : extended) : extended');
16550:  Function RadtoDeg( r : extended) : extended');
16551:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16552:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16553:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16554:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16555:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16556:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16557:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16558:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16559:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16560: end;
16561:
16562:
16563: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16564: begin
16565:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16566:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16567:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16568:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16569:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16570:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16571:    +'arth : extended; Phase : extended; end');
16572:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16573:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16574:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16575:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16576:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16577:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16578:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16579:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16580:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16581:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentricLonLat : TRpoint; RadiusVector :
       extended; UncorrectedEarthDistance : extended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended;
       ApparentRaDecl:TRPoint; end');
16582:   SIRegister_TAstronomy(CL);
16583:  Function AngleToStr( angle : extended) : string');
16584:  Function StrToAngle( s : string; var angle : extended) : boolean');
16585:  Function HoursToStr24( t : extended) : string');
16586:  Function RPoint( x, y : extended) : TRPoint');
16587:  Function getStimename( t : TDTType) : string');
16588: end;
16589:
16590: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16591: begin
16592:   TCardValue', 'Integer');
16593:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16594:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16595:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16596:   SIRegister_TCard(CL);
16597:   SIRegister_TDeck(CL);
16598: end;
16599:
```

```
16600: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16601: begin
16602:   tMethodCall', 'Procedure');
16603:   tVerboseCall', 'Procedure ( s : string)');
16604:  // PTEdge', '^TEdge // will not work');
16605:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16606:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16607:   SIRegister_TNode(CL);
16608:   SIRegister_TGraphList(CL);
16609: end;
16610:
16611: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16612: begin
16613:   ParserFloat', 'extended');
16614:   //PParserFloat', '^ParserFloat // will not work');
16615:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16616:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16617:   //POperation', '^TOperation // will not work');
16618:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16619:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16620:    +'; Token : TDFFToken; end');
16621:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16622:   (CL.FindClass('TOBJECT'),'EMathParserError');
16623:   CL.FindClass('TOBJECT'),'ESyntaxError');
16624:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16625:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16626:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16627:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16628:   (CL.FindClass('TOBJECT'),'EBadName');
16629:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16630:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16631:   SIRegister_TCustomParser(CL);
16632:   SIRegister_TExParser(CL);
16633: end;
16634:
16635:   function isService: boolean;
16636:   begin
16637:     result:= NOT(Application is TApplication);
16638:     {result:= Application is TServiceApplication;}
16639:   end;
16640:   function isApplication: boolean;
16641:   begin
16642:     result:= Application is TApplication;
16643:   end;
16644:   //SM_REMOTESESSION = $1000
16645:   function isTerminalSession: boolean;
16646:   begin
16647:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16648:   end;
16649:
16650: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16651: begin
16652:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16653:    +'String; margin_bottom : String; margin_left : String; margin_right : Strin'
16654:    +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16655:  Function cyURLEncode( const S : string) : string');
16656:  Function MakeResourceURL(const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16657:  Function MakeResourceURL1(const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;;
16658:  Function cyColorToHtml( aColor : TColor) : String');
16659:  Function HtmlToColor( aHtmlColor : String) : TColor');
16660:  //Function GetStreamEncoding( aStream : TStream) : TEncoding');
16661:  // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding) : Boolean');
16662:  Function AddHtmlUnicodePrefix( aHtml : String) : String');
16663:  Function RemoveHtmlUnicodePrefix( aHtml : String) : String');
16664:  Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup)');
16665:  Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup)');
16666:  CL.AddConstantN('IEBodyBorderless','String').SetString( 'none');
16667:  CL.AddConstantN('IEBodySingleBorder','String').SetString( '');
16668:  CL.AddConstantN('IEDesignModeOn','String').SetString( 'On');
16669:  CL.AddConstantN('IEDesignModeOff','String').SetString( 'Off');
16670:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF);
16671:  CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE);
16672: end;
16673:
16674:
16675: procedure SIRegister_UcomboV2(CL: TPSPascalCompiler);
16676: begin
16677:  CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600);
16678:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe'
16679:    +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe'
16680:    +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb'
16681:    +'inationsRepeat, CombinationsRepeatDown )');
16682:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16683:   SIRegister_TComboSet(CL);
16684: end;
16685:
16686:  procedure SIRegister_cyBaseComm(CL: TPSPascalCompiler);
16687: begin
16688:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )');
```

```
16689:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )');
16690:   TProcOnReceiveCommand', 'Procedure (Sender: TObject; aCommand: Word; userParam : Integer)');
16691:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo'
16692:    +'mmHandle : THandle; aString : String; userParam : Integer)');
16693:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from'
16694:    +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer)');
16695:   SIRegister_TcyBaseComm(CL);
16696:  CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1);
16697:  CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99);
16698:  Function ValidateFileMappingName( aName : String) : String');
16699:  procedure makeCaption(leftSide, Rightside:string; form:TForm);
16700: end;
16701:
16702: procedure SIRegister_cyDERUtils(CL: TPSPascalCompiler);
16703: begin
16704:   CL.AddTypeS('DERString', 'String');
16705:   CL.AddTypeS('DERChar', 'Char');
16706:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt'
16707:    +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16708:   CL.AddTypeS('TElementsTypes', 'set of TElementsType');
16709:   CL.AddTypeS('DERNString', 'String');
16710:  const DERDecimalSeparator','String').SetString( '.');
16711:  const DERDefaultChars','String')('+º@/%-
      _.:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16712:  const DERNDefaultChars','String').SetString( '/%-.0123456789abcdefghjkmnopqrstuvwxyz');
16713:  CL.AddDelphiFunction('Function isValidWebSiteChar( aChar : Char) : Boolean');
16714:  Function isValidWebMailChar( aChar : Char) : Boolean');
16715:  Function isValidWebSite( aStr : String) : Boolean');
16716:  Function isValidWebMail( aStr : String) : Boolean');
16717:  Function ValidateDate( aDERStr : DERString; var RsltFormat : String) : Boolean');
16718:  Function DERStrToDate( aDERStr, aFormat : String) : TDate');
16719:  Function IsDERChar( aChar : Char) : Boolean');
16720:  Function IsDERDefaultChar( aChar : Char) : Boolean');
16721:  Function IsDERMoneyChar( aChar : Char) : Boolean');
16722:  Function IsDERExceptionCar( aChar : Char) : Boolean');
16723:  Function IsDERSymbols( aDERString : String) : Boolean');
16724:  Function StringToDERCharSet( aStr : String) : DERString');
16725:  Function IsDERNDefaultChar( aChar : Char) : Boolean');
16726:  Function IsDERNChar( aChar : Char) : Boolean');
16727:  Function DERToDERNCharset( aDERStr : DERString) : DERNString');
16728:  Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean) : String');
16729:  Function DERExtractWebMail( aDERStr : DERString) : String');
16730:  Function DERExtractPhoneNr( aDERStr : DERString) : String');
16731:  Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16732:  Function DERExecute1(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
      RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
      SmartWebsiteRecognition:Boolean):TElementsType;
16733:  Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
      aElementsType : TElementsType) : String;');
16734:  Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
      Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType);');
16735: end;
16736:
16737: procedure SIRegister_cyImage(CL: TPSPascalCompiler);
16738: begin
16739:   pRGBQuadArray', '^TRGBQuadArray // will not work');
16740:  Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16741:  Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16742:  Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16743:  Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16744:  Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16745:  Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16746:  Procedure BitmapModifyRGB(Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool;
16747:  Procedure BitmapReplaceColor(Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
      RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16748:  Procedure BitmapReplaceColor1(Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
      PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
      PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16749:  Procedure BitmapReplaceColors(Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
      SingleDestinationColor, RefreshBmp : Boolean)');
16750:  Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16751:  Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16752:  Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16753:  Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16754:  Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean);');
16755:  Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16756:  Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16757:  Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)');
16758: end;
16759:
16760: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16761: begin
16762:   TMS2StrFormat', '( msHMSh, msHMS, msMSh, msMS, msSh, msS, msAh,msA )');
16763:   TPCMChannel', '( cMono, cStereo )');
16764:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16765:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16766:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono1'
16767:    +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
```

```
16768:    +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16769:    +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16770:    +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16771:    +'it48000Hz, Stereo16bit48000Hz )');
16772:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16773:    +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16774:    tWaveFormatEx', 'PWaveFormatEx');
16775:    HMMIO', 'Integer');
16776:   TWaveDeviceFormats', 'set of TPCMFormat');
16777:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16778:    +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16779:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16780:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16781:   TWaveOutOptions', 'set of TWaveOutOption');
16782:   TStreamOwnership2', '( soReference, soOwned )');
16783:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16784:  // PRawWave', '^TRawWave // will not work');
16785:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16786:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioError');
16787:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioSysError');
16788:   CL.AddClassN(CL.FindClass('TOBJECT'),'EWaveAudioInvalidOperation');
16789:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16790:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var pW'
16791:    +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16792:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16793:    +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16794:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16795:    +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16796:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16797:    +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16798:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16799:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16800:  GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16801:  CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16802:  CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16803:  GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16804:  CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16805:  OpenStreamWaveAudio( Stream : TStream) : HMMIO');
16806:  CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16807:  GetAudioFormat( FormatTag : Word) : String');
16808:  GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
16809:  GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16810:  GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16811:  GetWaveAudioPeakLevel(const Data: TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16812:  InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16813:  SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16814:  ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16815:  MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
       TObject; BufferSize : DWORD) : Boolean');
16816:  ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
       dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16817:  SetPCMAudioFormat(const pWaveFormat: PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
       TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16818:  Procedure SetPCMAudioFormatS( const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16819:  GetPCMAudioFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16820:  GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');
16821:  MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat) : String');
16822:  WaitForSyncObject( SyncObject : THandle; Timeout : DWORD) : DWORD');
16823:  //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT');
16824: end;
16825:
16826: procedure SIRegister_NamedPipes(CL: TPSPascalCompiler);
16827: begin
16828:  'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096);
16829:  CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000);
16830:  'PIPE_NAMING_SCHEME','String').SetString( '\\%s\pipe\%s');
16831:  'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFFFF ));
16832:  'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1);
16833:  'STATUS_SUCCESS','LongWord').SetUInt( $00000000);
16834:  'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005);
16835:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16836:   CL.AddTypeS('TPipeType', '( ptyp_ByteByte, ptyp_MsgByte, ptyp_MsgMsg )');
16837:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16838:   SIRegister_TNamedPipe(CL);
16839:   SIRegister_TServerPipe(CL);
16840:   SIRegister_TClientPipe(CL);
16841:  CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD) : DWORD');
16842:  Function HasOverlappedIoCompleted( const ov : OVERLAPPED) : Boolean');
16843:  Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean):
       TOverlappedResult;
16844:  Function GetStreamAsText( stm : TStream) : string');
16845:  Procedure SetStreamAsText( const aTxt : string; stm : TStream)');
16846: end;
16847:
16848: procedure SIRegister_DPUtils(CL: TPSPascalCompiler);
16849: begin
16850:  // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16851:   SIRegister_TThumbData(CL);
16852:  'PIC_BMP','LongInt').SetInt( 0);
```

```
16853:  'PIC_JPG','LongInt').SetInt( 1);
16854:  'THUMB_WIDTH','LongInt').SetInt( 60);
16855:  'THUMB_HEIGHT','LongInt').SetInt( 60);
16856:  Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean');
16857:  Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)');
16858:  Function ReadBitmap( Filehandle : Integer; Width, Height : Integer) : TBitmap');
16859:  Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap)');
16860:  Function OpenPicture( fn : string; var tp : Integer) : Integer');
16861:  Function ConvertPicture( pi : Integer; tp : Integer) : TBitmap');
16862:  Function LoadPicture( fn : string; var w, h : Integer) : TBitmap');
16863:  Function TurnBitmap( bmp : TBitmap; ang : Integer) : TBitmap');
16864:  Function RotateBitmap( Bitmap : TBitmap; Direction : Integer) : TBitmap');
16865:  Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap) : TRect');
16866:  Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer) : TBitmap');
16867:  Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer)');
16868:  Procedure FindFiles( path, mask : string; items : TStringList)');
16869:  Function LetFileName( s : string) : string');
16870:  Function LetParentPath( path : string) : string');
16871:  Function AddBackSlash( path : string) : string');
16872:  Function CutBackSlash( path : string) : string');
16873: end;
16874:
16875: procedure SIRegister_CommonTools(CL: TPSPascalCompiler);
16876: begin
16877:  //'BYTES','LongInt').SetInt( 1);
16878:  'KBYTES','LongInt').SetInt( 1024);
16879:  'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABE11 ));
16880:  'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ));
16881:  'DBG_GONE','LongWord').SetUInt( $99AC1D99);
16882:  'SHELL_NS_MYCOMPUTER','String').SetString( '::{20D04FE0-3AEA-1069-A2D8-08002B30309D}');
16883:   SIRegister_MakeComServerMethodsPublic(CL);
16884:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16885:  Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD) : Boolean');
16886:  Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean)');
16887:  Function TBGetTempFolder : string');
16888:  Function TBGetTempFile : string');
16889:  Function TBGetModuleFilename : string');
16890:  Function FormatModuleVersionInfo( const aFilename : string) : string');
16891:  Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD) : string');
16892:  Function TBGetFileSize( aFile : string; aMultipleOf : Integer) : Integer');
16893:  Function FormatAttribString( aAttr : Integer) : string');
16894:  Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo) : string');
16895:  Function ShellRecycle( aWnd : HWND; aFileOrFolder : string) : Boolean');
16896:  Function IsDebuggerPresent : BOOL');
16897:  Function TBNotImplemented : HRESULT');
16898: end;
16899:
16900: procedure SIRegister_D2_VistaHelperU(CL: TPSPascalCompiler);
16901: begin
16902:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16903:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16904:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16905:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean;');
16906:   //TDrivesProperty = array['A'..'Z'] of boolean;
16907:  Function TBSetSystemTime( DateTime : TDateTime; DOW : word) : boolean');
16908:  Function IsElevated : Boolean');
16909:  Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:TGUID;const aIID:TGUID;out aObj:TObject);
16910:   CL.AddTypeS('TPasswordUsage', '( pu_None, pu_Default, pu_Defined )');
16911:  Function TrimNetResource( UNC : string) : string');
16912:  Procedure GetFreeDrives( var FreeDrives : TDrivesProperty)');
16913:  Procedure GetMappedDrives( var MappedDrives : TDrivesProperty)');
16914:  Function MapDrive(UNCPath:string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
       UserUsage : TPasswordUsage; User : string; Comment : string) : boolean');
16915:  Function UnmapDrive( Drive : char; Force : boolean) : boolean');
16916:  Function TBIsWindowsVista : Boolean');
16917:  Procedure SetVistaFonts( const AForm : TForm)');
16918:  Procedure SetVistaContentFonts( const AFont : TFont)');
16919:  Function GetProductType( var sType : String) : Boolean');
16920:  Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16921:  Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16922:  Function lstrcpyn( lpString1, lpString2 : PChar; iMaxLength : Integer) : PChar');
16923:  Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16924:  Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16925:  Function lstrlen( lpString : PChar) : Integer');
16926:  Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16927:  Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
       TokenInformation : ___Pointer; TokenInformationLength : DWORD) : BOOL');
16928: end;
16929:
16930: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16931: begin
16932:  'cLineTerminator','Char').SetString( #10);
16933:  'cLineTerminators','String').SetString( #13#10);
16934:  'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD ( - 1 ));
16935:   SIRegister_TFixedCriticalSection(CL);
16936:   SIRegister_TMultiReadSingleWrite(CL);
16937:  CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16938:  Function GetDecimalSeparator : Char');
```

```
16939:  TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16940:  Procedure CollectFiles(const directory,fileMask:UnicodeString; list:TStrings;
       recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16941:   CL.AddTypeS('NativeInt', 'Integer');
16942:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16943:   CL.AddTypeS('NativeUInt', 'Cardinal');
16944:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16945:   //CL.AddTypeS('TBytes', 'array of Byte');
16946:   CL.AddTypeS('RawByteString', 'UnicodeString');
16947:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16948:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16949:   SIRegister_TPath(CL);
16950:   SIRegister_TFile(CL);
16951:   SIRegister_TdwsThread(CL);
16952:  Function GetSystemMilliseconds : Int64');
16953:  Function UTCDateTime : TDateTime');
16954:  Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16955:  Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16956:  Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16957:  Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
16958:  Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer;');
16959:  Function UnicodeComparePChars1( p1, p2 : PChar; n : Integer) : Integer;');
16960:  Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16961:  Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16962:  Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16963:  Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16964:  Function InterlockedIncrement( var val : Integer) : Integer');
16965:  Function InterlockedDecrement( var val : Integer) : Integer');
16966:  Procedure FastInterlockedIncrement( var val : Integer)');
16967:  Procedure FastInterlockedDecrement( var val : Integer)');
16968:  Function InterlockedExchangePointer( var target : ___Pointer; val : ___Pointer) : ___Pointer');
16969:  Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16970:  Procedure dwsOutputDebugString( const msg : UnicodeString)');
16971:  Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:Str);
16972:  Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16973:  Procedure VarCopy( out dest : Variant; const src : Variant)');
16974:  Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16975:  Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16976:  Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16977:  Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16978:  Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
16979:  Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
16980:  Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
16981:  Procedure CloseFileHandle( hFile : THandle)');
16982:  Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
16983:  Function FileMove( const existing, new : UnicodeString) : Boolean');
16984:  Function dwsFileDelete( const fileName : String) : Boolean');
16985:  Function FileRename( const oldName, newName : String) : Boolean');
16986:  Function dwsFileSize( const name : String) : Int64');
16987:  Function dwsFileDateTime( const name : String) : TDateTime');
16988:  Function DirectSet8087CW( newValue : Word) : Word');
16989:  Function DirectSetMXCSR( newValue : Word) : Word');
16990:  Function TtoObject( const T: byte) : TObject');
16991:  Function TtoPointer( const T: byte) : ___Pointer');
16992:  Procedure GetMemForT(var T: byte; Size : integer)');
16993:  Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');
16994: end;
16995:
16996: procedure SIRegister_AdSocket(CL: TPSPascalCompiler);
16997: begin
16998:  'IPStrSize','LongInt').SetInt( 15);
16999:  'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711);
17000:  'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712);
17001:  'ADWSBASE','LongInt').SetInt( 9000);
17002:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
17003:    +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
17004:   SIRegister_EApdSocketException(CL);
17005:   TWsMode', '( wsClient, wsServer )');
17006:   TWsNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket');
17007:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrCode : Integer)');
17008:   SIRegister_TApdSocket(CL);
17009: end;
17010:
17011: procedure SIRegister_AdPort(CL: TPSPascalCompiler);
17012: begin
17013:   SIRegister_TApdCustomComPort(CL);
17014:   SIRegister_TApdComPort(CL);
17015:  Function ComName( const ComNumber : Word) : ShortString');
17016:  Function SearchComPort( const C : TComponent) : TApdCustomComPort');
17017: end;
17018:
17019: procedure SIRegister_PathFunc(CL: TPSPascalCompiler);
17020: begin
17021:  Function inAddBackslash( const S : String) : String');
17022:  Function PathChangeExt( const Filename, Extension : String) : String');
17023:  Function PathCharCompare( const S1, S2 : PChar) : Boolean');
17024:  Function PathCharIsSlash( const C : Char) : Boolean');
17025:  Function PathCharIsTrailByte( const S : String; const Index : Integer) : Boolean');
17026:  Function PathCharLength( const S : String; const Index : Integer) : Integer');
```

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

```
17113:  Function IsAdminLoggedOn : Boolean');
17114:  Function IsPowerUserLoggedOn : Boolean');
17115:  Function IsMultiByteString( const S : AnsiString) : Boolean');
17116:  Function FontExists( const FaceName : String) : Boolean');
17117:  //Procedure FreeAndNil( var Obj)');
17118:  Function SafeLoadLibrary( const Filename : String; ErrorMode : UINT) : HMODULE');
17119:  Function GetUILanguage : LANGID');
17120:  Function RemoveAccelChar( const S : String) : String');
17121:  Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17122:  Function AddPeriod( const S : String) : String');
17123:  Function GetExceptMessage : String');
17124:  Function GetPreferredUIFont : String');
17125:  Function IsWildcard( const Pattern : String) : Boolean');
17126:  Function WildcardMatch( const Text, Pattern : PChar) : Boolean');
17127:  Function IntMax( const A, B : Integer) : Integer');
17128:  Function Win32ErrorString( ErrorCode : Integer) : String');
17129:  Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet)');
17130:  Function inCompareMem( P1, P2 : TObject; Length : Integer) : Boolean');
17131:  Function DeleteDirTree( const Dir : String) : Boolean');
17132:  Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17133:  Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT)');
17134:  // CL.AddTypeS('TSysCharSet', 'set of AnsiChar');
17135:  Function inCharInSet( C : Char; const CharSet : TSysCharSet) : Boolean');
17136:  Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String) : Boolean');
17137:  Function ShutdownBlockReasonDestroy( Wnd : HWND) : Boolean');
17138:  Function TryStrToBoolean( const S : String; var BoolResult : Boolean) : Boolean');
17139:  Procedure WaitMessageWithTimeout( const Milliseconds : DWORD)');
17140:  Function MoveFileReplace(const ExistingFileName, NewFileName : String) : Boolean');
17141:  Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND)');
17142: end;
17143:
17144: procedure SIRegister_CmnFunc(CL: TPSPascalCompiler);
17145: begin
17146:   SIRegister_TWindowDisabler(CL);
17147:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17148:   TMsgBoxCallbackFunc','procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17149:  Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox)');
17150:  Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17151:  Function AppMessageBox( const Text, Caption : PChar; Flags : Longint) : Integer');
17152:  Function MsgBoxP( const Text,Caption: PChar; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17153:  Function inMsgBox(const Text,Caption:String; const Typ: TMsgBoxType;const Buttons:Cardinal):Int;
17154:  Function MsgBoxFmt(const Text:String;const Args:array of const;const Caption:String;const
       Typ:TMsgBoxType;const Buttons:Cardinal):Integer');
17155:  Procedure ReactivateTopWindow');
17156:  Procedure SetMessageBoxCaption( const Typ : TMsgBoxType; const NewCaption : PChar)');
17157:  Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean)');
17158:  Procedure SetMessageBoxCallbackFunc(const AFunc : TMsgBoxCallbackFunc; const AParam : LongInt)');
17159: end;
17160:
17161: procedure SIRegister_ImageGrabber(CL: TPSPascalCompiler);
17162: begin
17163:   SIRegister_TImageGrabber(CL);
17164:   SIRegister_TCaptureDrivers(CL);
17165:   SIRegister_TCaptureDriver(CL);
17166: end;
17167:
17168: procedure SIRegister_SecurityFunc(CL: TPSPascalCompiler);
17169: begin
17170:  Function GrantPermissionOnFile(const DisableFsRedir:Boolean; Filename:String;const
       Entries:TGrantPermissionEntry; const EntryCount:Integer):Boolean');
17171:  Function GrantPermissionOnKey(const RegView:TRegView;const RootKey:HKEY;const Subkey:String;const
       Entries: TGrantPermissionEntry; const EntryCount:Integer): Boolean');
17172: end;
17173:
17174: procedure SIRegister_RedirFunc(CL: TPSPascalCompiler);
17175: begin
17176:   CL.AddTypeS('TPreviousFsRedirectionState', record DidDisable : Boolean; OldValue : ___Pointer; end');
17177:  CL.AddDelphiFunction('Function AreFsRedirectionFunctionsAvailable : Boolean');
17178:  Function DisableFsRedirectionIf(const Disable:Boolean;var PreviousState:TPreviousFsRedirectionState):Boo;
17179:  Procedure RestoreFsRedirection( const PreviousState : TPreviousFsRedirectionState)');
17180:  Function CreateDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17181:  Function CreateProcessRedir(const DisableFsRedir:Boolean;const lpApplicationName:PChar;const
       lpCommandLine:PChar; const lpProcessAttributes, lpThreadAttributes:PSecurityAttributes;const
       bInheritHandles:BOOL; const dwCreationFlags:DWORD;const lpEnvironment:Pointer; const
       lpCurrentDirectory:PChar; const lpStartupInfo : TStartupInfo; var
       lpProcessInformation:TProcessInformation):BOOL;
17182:  Function CopyFileRedir(const DisableFsRedir:Boolean; const ExistingFilename, NewFilename : String; const
       FailIfExists : BOOL) : BOOL');
17183:  Function DeleteFileRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17184:  Function DirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17185:  Function FileOrDirExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17186:  Function FindFirstFileRedir( const DisableFsRedir : Boolean; const Filename : String; var FindData :
       TWin32FindData) : THandle');
17187:  Function GetFileAttributesRedir( const DisableFsRedir : Boolean; const Filename : String) : DWORD');
17188:  Function GetShortNameRedir( const DisableFsRedir : Boolean; const Filename : String) : String');
17189:  Function GetVersionNumbersRedir(const DisableFsRedir:Boolean; const Filename:String; var VersionNumbers :
       TFileVersionNumbers) : Boolean');
17190:  Function IsDirectoryAndNotReparsePointRedir(const DisableFsRedir:Boolean;const Name:String):Bool;
17191:  Function MoveFileRedir(const DisableFsRedir:Bool;const ExistingFilename,NewFilename:String):BOOL;
```

```
17192:  Function MoveFileExRedir(const DisableFsRedir:Boolean;const ExistingFilename,NewFilename:String;const
       Flags:DWORD):BOOL;
17193:  Function NewFileExistsRedir( const DisableFsRedir : Boolean; const Filename : String) : Boolean');
17194:  Function RemoveDirectoryRedir( const DisableFsRedir : Boolean; const Filename : String) : BOOL');
17195:  Function SetFileAttributesRedir(const DisableFsRedir:Bool;const Filename:String;const Attrib:DWORD):BOOL;
17196:  Function SetNTFSCompressionRedir(const DisableFsRedir:Boolean;const FileOrDir:String;Compress:Bool:Bool;
17197:   SIRegister_TFileRedir(CL);
17198:   SIRegister_TTextFileReaderRedir(CL);
17199:   SIRegister_TTextFileWriterRedir(CL);
17200: end;
17201:
17202: procedure SIRegister_Int64Em(CL: TPSPascalCompiler);
17203: begin
17204:   //CL.AddTypeS('LongWord', 'Cardinal');
17205:   CL.AddTypeS('Integer64', 'record Lo : LongWord; Hi : LongWord; end');
17206:  Function Compare64( const N1, N2 : Integer64) : Integer');
17207:  Procedure Dec64( var X : Integer64; N : LongWord)');
17208:  Procedure Dec6464( var X : Integer64; const N : Integer64)');
17209:  Function Div64( var X : Integer64; const Divisor : LongWord) : LongWord');
17210:  Function Inc64( var X : Integer64; N : LongWord) : Boolean');
17211:  Function Inc6464( var X : Integer64; const N : Integer64) : Boolean');
17212:  Function Integer64ToStr( X : Integer64) : String');
17213:  Function Mod64( const X : Integer64; const Divisor : LongWord) : LongWord');
17214:  Function Mul64( var X : Integer64; N : LongWord) : Boolean');
17215:  Procedure Multiply32x32to64( N1, N2 : LongWord; var X : Integer64)');
17216:  Procedure Shr64( var X : Integer64; Count : LongWord)');
17217:  Function StrToInteger64( const S : String; var X : Integer64) : Boolean');
17218: end;
17219:
17220: procedure SIRegister_InstFunc(CL: TPSPascalCompiler);
17221: begin
17222:   //CL.AddTypeS('PSimpleStringListArray', '^TSimpleStringListArray // will not work');
17223:   SIRegister_TSimpleStringList(CL);
17224:   CL.AddTypeS('TExecWait', '( ewNoWait, ewWaitUntilTerminated, ewWaitUntilIdle)');
17225:   CL.AddTypeS('TDetermineDefaultLanguageResult', '( ddNoMatch, ddMatch, ddMatchLangParameter )');
17226:   CL.AddTypeS('TMD5Digest', 'array[0..15] of Byte;');
17227:   CL.AddTypeS('TSHA1Digest', 'array[0..19] of Byte;');
17228:  // TMD5Digest = array[0..15] of Byte;
17229:   //     TSHA1Digest = array[0..19] of Byte;
17230:  Function CheckForMutexes( Mutexes : String) : Boolean');
17231:  Function CreateTempDir : String');
17232:  Function DecrementSharedCount( const RegView : TRegView; const Filename : String) : Boolean');
17233:  Procedure DelayDeleteFile( const DisableFsRedir : Boolean; const Filename : String; const MaxTries,
       FirstRetryDelayMS, SubsequentRetryDelayMS : Integer)');
17234:  //Function DelTree( const DisableFsRedir : Boolean; const Path : String; const IsDir, DeleteFiles,
       DeleteSubdirsAlso, BreakOnError : Boolean; const DeleteDirProc : TDeleteDirProc; const DeleteFileProc :
       TDeleteFileProc; const Param : Pointer) : Boolean');
17235:  //Function DetermineDefaultLanguage( const GetLanguageEntryProc : TGetLanguageEntryProc; const Method :
       TSetupLanguageDetectionMethod; const LangParameter : String; var ResultIndex : Integer) :
       TDetermineDefaultLanguageResult');
17236:  //Procedure EnumFileReplaceOperationsFilenames(const EnumFunc:TEnumFROFilenamesProc;Param:Pointer);
17237:  Function GenerateNonRandomUniqueFilename( Path : String; var Filename : String) : Boolean');
17238:  Function GenerateUniqueName(const DisableFsRedir:Bool;Path:String;const Extension:String):String;
17239:  Function GetComputerNameString : String');
17240:  Function GetFileDateTime(const DisableFsRedir:Boolean;const Filename:String;var DateTime:TFileTime):Bool;
17241:  Function GetMD5OfFile( const DisableFsRedir : Boolean; const Filename : String) : TMD5Digest');
17242:  Function GetMD5OfAnsiString( const S : AnsiString) : TMD5Digest');
17243: // Function GetMD5OfUnicodeString( const S : UnicodeString) : TMD5Digest');
17244:  Function GetSHA1OfFile(const DisableFsRedir:Boolean;const Filename:String):TSHA1Digest');
17245:  Function GetSHA1OfAnsiString( const S : AnsiString) : TSHA1Digest');
17246: // Function GetSHA1OfUnicodeString( const S : UnicodeString) : TSHA1Digest');
17247:  Function GetRegRootKeyName( const RootKey : HKEY) : String');
17248:  Function GetSpaceOnDisk(const DisableFsRedir:Bool;const DriveRoot:String;var FreeBytes,
       TotBytes:Int64):Bool;
17249:  Function GetSpaceOnNearestMountPoint(const DisableFsRedir:Bool;const StartDir:String;var FreeBytes,
       TotalBytes: Integer64):Bool;
17250:  Function GetUserNameString : String');
17251:  Procedure IncrementSharedCount(const RegView:TRegView;const Filename:String;const AlreadyExisted:Boolean);
17252:  //Function InstExec( const DisableFsRedir : Boolean; const Filename, Params : String; WorkingDir :
       String; const Wait:TExecWait;const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var
       ResultCode:Integer) : Boolean');
17253: // Function InstShellExec( const Verb, Filename, Params : String; WorkingDir : String; const Wait :
       TExecWait; const ShowCmd:Integer;const ProcessMessagesProc:TProcedure;var ResultCode:Integer):Boolean;
17254:  Procedure InternalError( const Id : String)');
17255:  Procedure InternalErrorFmt( const S : String; const Args : array of const)');
17256:  Function IsDirEmpty( const DisableFsRedir : Boolean; const Dir : String) : Boolean');
17257:  Function IsProtectedSystemFile(const DisableFsRedir:Boolean; const Filename:String) : Boolean');
17258:  Function MakePendingFileRenameOperationsChecksum : TMD5Digest');
17259:  Function ModifyPifFile( const Filename : String; const CloseOnExit : Boolean) : Boolean');
17260:  Procedure RaiseFunctionFailedError( const FunctionName : String)');
17261:  Procedure RaiseOleError( const FunctionName : String; const ResultCode : HRESULT)');
17262:  Procedure RefreshEnvironment');
17263:  Function ReplaceSystemDirWithSysWow64( const Path : String) : String');
17264:  Function ReplaceSystemDirWithSysNative( Path : String; const IsWin64 : Boolean) : String');
17265:  Procedure UnregisterFont( const FontName, FontFilename : String)');
17266:  Function RestartComputer : Boolean');
17267:  Procedure RestartReplace( const DisableFsRedir : Boolean; TempFile, DestFile : String)');
17268:  Procedure SplitNewParamStr( const Index : Integer; var AName, AValue : String)');
17269:  Procedure Win32ErrorMsg( const FunctionName : String)');
```

```
17270:  Procedure Win32ErrorMsgEx( const FunctionName : String; const ErrorCode : DWORD)');
17271:  Function inForceDirectories(const DisableFsRedir:Boolean; Dir : String) : Boolean');
17272:  //from Func2
17273:  //Function inCreateShellLink( const Filename, Description, ShortcutTo, Parameters, WorkingDir : String;
       IconFilename : String; const IconIndex, ShowCmd : Integer; const HotKey : Word; FolderShortcut : Boolean; '
17274:    //+'const AppUserModelID:String;const ExcludeFromShowInNewInstall,PreventPinning:Bool):String');
17275:  Procedure RegisterTypeLibrary( const Filename : String)');
17276:  //Procedure UnregisterTypeLibrary( const Filename : String)');
17277:  //Function UnpinShellLink( const Filename : String) : Boolean');}
17278:  function getVersionInfoEx3: TOSVersionInfoEx;');
17279:  Function GetVersionEx3(out verinfo: TOSVersionInfoEx): boolean;');
17280:  procedure InitOle;');
17281:  Function ExpandConst( const S : String) : String');
17282:  Function ExpandConstEx( const S : String; const CustomConsts : array of String) : String');
17283:  Function ExpandConstEx2(const S:Str;const CustConsts:array of Str;const DoExpandIndividualConst:Bool):Str;
17284:  Function ExpandConstIfPrefixed( const S : String) : String');
17285:  Procedure LogWindowsVersion');
17286:  Function EvalCheck( const Expression : String) : Boolean');
17287: end;
17288:
17289: procedure SIRegister_unitResourceDetails(CL: TPSPascalCompiler);
17290: begin
17291:   CL.AddClassN(CL.FindClass('TOBJECT'),'TResourceDetails');
17292:   //CL.AddTypeS('TResourceDetailsClass', 'class of TResourceDetails');
17293:   SIRegister_TResourceModule(CL);
17294:   SIRegister_TResourceDetails(CL);
17295:   SIRegister_TAnsiResourceDetails(CL);
17296:   SIRegister_TUnicodeResourceDetails(CL);
17297:  Procedure RegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17298:  Procedure UnRegisterResourceDetails( resourceClass : TResourceDetailsClass)');
17299:  Function ResourceWideCharToStr( var wstr : PWideChar; codePage : Integer) : string');
17300:  Procedure ResourceStrToWideChar( const s : string; var p : PWideChar; codePage : Integer)');
17301:  Function ResourceNameToInt( const s : string) : Integer');
17302:  Function CompareDetails( p1, p2 : TObject(Pointer)) : Integer');
17303: end;
17304:
17305:
17306: procedure SIRegister_TSimpleComPort(CL: TPSPascalCompiler);
17307: begin
17308:   //with RegClassS(CL,'TObject', 'TSimpleComPort') do
17309:   with CL.AddClassN(CL.FindClass('TObject'),'TSimpleComPort') do begin
17310:     RegisterMethod('Constructor Create');
17311:     RegisterMethod('Procedure Free');
17312:     RegisterMethod('Procedure Open( PortNumber : Integer; const Parameters : String)');
17313:     RegisterMethod('Procedure WriteString( const S : String)');
17314:     RegisterMethod('Procedure ReadString( var S : String)');
17315:   end;
17316:  Ex.: SimpleComPort:= TSimpleComPort.Create;
17317:       SimpleComPort.Open(1, 'baud=115200 parity=N data=8 stop=1');
17318:       SimpleComPort.WriteString(AsciiChar);
17319: end;
17320:
17321:
17322: procedure SIRegister_Console(CL: TPSPascalCompiler);
17323: begin
17324:  CL.AddConstantN('White','LongInt').SetInt( 15); }
17325: // CL.AddConstantN('Blink','LongInt').SetInt( 128);
17326:  CL.AddConstantN('conBW40','LongInt').SetInt( 0);
17327:  CL.AddConstantN('conCO40','LongInt').SetInt( 1);
17328:  CL.AddConstantN('conBW80','LongInt').SetInt( 2);
17329:  CL.AddConstantN('conCO80','LongInt').SetInt( 3);
17330:  CL.AddConstantN('conMono','LongInt').SetInt( 7);
17331:  CL.AddConstantN('conFont8x8','LongInt').SetInt( 256);
17332:  //CL.AddConstantN('C40','').SetString( CO40);
17333:  //CL.AddConstantN('C80','').SetString( CO80);
17334:  Function conReadKey : Char');
17335:  Function conKeyPressed : Boolean');
17336:  Procedure conGotoXY( X, Y : Smallint)');
17337:  Function conWhereX : Integer');
17338:  Function conWhereY : Integer');
17339:  Procedure conTextColor( Color : Byte);');
17340:  Function conTextColor1 : Byte;');
17341:  Procedure conTextBackground( Color : Byte);');
17342:  Function conTextBackground1 : Byte;');
17343:  Procedure conTextMode( Mode : Word)');
17344:  Procedure conLowVideo');
17345:  Procedure conHighVideo');
17346:  Procedure conNormVideo');
17347:  Procedure conClrScr');
17348:  Procedure conClrEol');
17349:  Procedure conInsLine');
17350:  Procedure conDelLine');
17351:  Procedure conWindow( Left, Top, Right, Bottom : Integer)');
17352:  Function conScreenWidth : Smallint');
17353:  Function conScreenHeight : Smallint');
17354:  Function conBufferWidth : Smallint');
17355:  Function conBufferHeight : Smallint');
17356:  procedure InitScreenMode;');
17357: end;
```

```
17358:
17359: (*----------------------------------------------------------------------------*)
17360: procedure SIRegister_testutils(CL: TPSPascalCompiler);
17361: begin
17362:   SIRegister_TNoRefCountObject(CL);
17363:  Procedure FreeObjects( List : TFPList)');
17364:  Procedure GetMethodList( AObject : TObject; AList : TStrings);');
17365:  Procedure GetMethodList1( AClass : TClass; AList : TStrings);');
17366: end;
17367:
17368: procedure SIRegister_ToolsUnit(CL: TPSPascalCompiler);
17369: begin
17370:  'MaxDataSet','LongInt').SetInt( 35);
17371:   //CL.AddTypeS('TDBConnectorClass', 'class of TDBConnector');
17372:   SIRegister_TDBConnector(CL);
17373:   SIRegister_TDBBasicsTestSetup(CL);
17374:   SIRegister_TTestDataLink(CL);
17375:  'testValuesCount','LongInt').SetInt( 25);
17376:  Procedure InitialiseDBConnector');
17377:  Procedure FreeDBConnector');
17378:  Function DateTimeToTimeString( d : tdatetime) : string');
17379:  Function TimeStringToDateTime( d : String) : TDateTime');
17380: end;
17381:
17382: procedure SIRegister_fpcunit(CL: TPSPascalCompiler);
17383: begin
17384:   SIRegister_EAssertionFailedError(CL);
17385:   CL.AddTypeS('TTestStep', '( stSetUp, stRunTest, stTearDown, stNothing )');
17386:   CL.AddTypeS('TRunMethod', 'Procedure');
17387:   CL.AddClassN(CL.FindClass('TOBJECT'),'TTestResult');
17388:   SIRegister_TTest(CL);
17389:   SIRegister_TAssert(CL);
17390:   SIRegister_TTestFailure(CL);
17391:   SIRegister_ITestListener(CL);
17392:   SIRegister_TTestCase(CL);
17393:   //CL.AddTypeS('TTestCaseClass', 'class of TTestCase');
17394:   SIRegister_TTestSuite(CL);
17395:   SIRegister_TTestResult(CL);
17396:  Function ComparisonMsg( const aExpected : string; const aActual : string) : string');
17397: end;
17398:
17399: procedure SIRegister_cTCPBuffer(CL: TPSPascalCompiler);
17400: begin
17401:   TOBJECT'),'ETCPBuffer');
17402:   TTCPBuffer', 'record Ptr : TObject; Size : Integer; Max : Intege'
17403:    +r; Head : Integer; Used : Integer; end');
17404:  'ETHERNET_MTU_100MBIT','LongInt').SetInt( 1500);
17405:  'ETHERNET_MTU_1GBIT','LongInt').SetInt( 9000);
17406:  'TCP_BUFFER_DEFAULTMAXSIZE','LongInt').SetInt( ETHERNET_MTU_1GBIT * 4);
17407:  'TCP_BUFFER_DEFAULTBUFSIZE','LongInt').SetInt( ETHERNET_MTU_100MBIT * 4);
17408:  Procedure TCPBufferInitialise(var TCPBuf:TTCPBuffer;const TCPBufMaxSize:Int;const TCPBufSize:Int;
17409:  Procedure TCPBufferFinalise( var TCPBuf : TTCPBuffer)');
17410:  Procedure TCPBufferPack( var TCPBuf : TTCPBuffer)');
17411:  Procedure TCPBufferResize( var TCPBuf : TTCPBuffer; const TCPBufSize : Integer)');
17412:  Procedure TCPBufferExpand( var TCPBuf : TTCPBuffer; const Size : Integer)');
17413:  Procedure TCPBufferShrink( var TCPBuf : TTCPBuffer)');
17414:  Function TCPBufferAddPtr( var TCPBuf : TTCPBuffer; const Size : Integer) : Pointer');
17415:  Procedure TCPBufferAddBuf( var TCPBuf : TTCPBuffer; const Buf : string; const Size : Integer)');
17416:  Function TCPBufferPeekPtr( const TCPBuf : TTCPBuffer; var BufPtr : Pointer) : Integer');
17417:  Function TCPBufferPeek( var TCPBuf : TTCPBuffer; var Buf: string; const Size:Integer): Integer');
17418:  Function TCPBufferRemove(var TCPBuf : TTCPBuffer;var Buf: string; const Size:Integer): Integer');
17419:  Procedure TCPBufferClear( var TCPBuf : TTCPBuffer)');
17420:  Function TCPBufferDiscard( var TCPBuf : TTCPBuffer; const Size : Integer) : Integer');
17421:  Function TCPBufferUsed( const TCPBuf : TTCPBuffer) : Integer');
17422:  Function TCPBufferEmpty( const TCPBuf : TTCPBuffer) : Boolean');
17423:  Function TCPBufferAvailable( const TCPBuf : TTCPBuffer) : Integer');
17424:  Function TCPBufferPtr( const TCPBuf : TTCPBuffer) : Pointer');
17425:  Procedure TCPBufferSetMaxSize( var TCPBuf : TTCPBuffer; const MaxSize : Integer)');
17426: end;
17427:
17428: procedure SIRegister_Glut(CL: TPSPascalCompiler);
17429: begin
17430:   //CL.AddTypeS('PInteger', '^Integer // will not work');
17431:   //CL.AddTypeS('PPChar', '^PChar // will not work');
17432:  CL.AddConstantN('GLUT_API_VERSION','LongInt').SetInt( 3);
17433:  CL.AddConstantN('GLUT_XLIB_IMPLEMENTATION','LongInt').SetInt( 12);
17434:  CL.AddConstantN('GLUT_RGB','LongInt').SetInt( 0);
17435:  CL.AddConstantN('GLUT_GAME_MODE_REFRESH_RATE','LongInt').SetInt( 5);
17436:  CL.AddConstantN('GLUT_GAME_MODE_DISPLAY_CHANGED','LongInt').SetInt( 6);
17437:  Procedure LoadGlut( const dll : String)');
17438:  Procedure FreeGlut');
17439: end;
17440:
17441: procedure SIRegister_LEDBitmaps(CL: TPSPascalCompiler);
17442: begin
17443:   CL.AddTypeS('TLEDColor', '( ledcGreen, ledcRed, ledcGray )');
17444:  Function GetLEDBitmapHandle( Color : TLEDColor; State : Boolean) : THandle');
17445: end;
17446:
```

```
17447: procedure SIRegister_SwitchLed(CL: TPSPascalCompiler);
17448: begin
17449:   TLedColor', '( Aqua, Pink, Purple, Red, Yellow, Green, Blue, Orange, Black )');
17450:   TTLedState', '( LedOn, LedOff, LedDisabled )');
17451:   SIRegister_TSwitchLed(CL);
17452:  //CL.AddDelphiFunction('Procedure Register');
17453: end;
17454:
17455: procedure SIRegister_FileClass(CL: TPSPascalCompiler);
17456: begin
17457:   CL.AddTypeS('TFileCreateDisposition', '( fdCreateAlways, fdCreateNew, fdOpenE'
17458:    +'xisting, fdOpenAlways, fdTruncateExisting )');
17459:   CL.AddTypeS('TIFileAccess', '( faRead, faWrite, faReadWrite )');
17460:   CL.AddTypeS('TIFileSharing', '( fsNone, fsRead, fsWrite, fsReadWrite )');
17461:   SIRegister_TCustomFile(CL);
17462:   SIRegister_TIFile(CL);
17463:   SIRegister_TMemoryFile(CL);
17464:   SIRegister_TTextFileReader(CL);
17465:   SIRegister_TTextFileWriter(CL);
17466:   SIRegister_TFileMapping(CL);
17467:   SIRegister_EFileError(CL);
17468: end;
17469:
17470: procedure SIRegister_FileUtilsClass(CL: TPSPascalCompiler);
17471: begin
17472:   CL.AddTypeS('TFileAttributes', '( ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID'
17473:    +', ffaDirectory, ffaArchive, ffaAnyFile )');
17474:   CL.AddTypeS('TAttributeSet', 'set of TFileAttributes');
17475:   SIRegister_TFileSearch(CL);
17476: end;
17477:
17478: procedure SIRegister_uColorFunctions(CL: TPSPascalCompiler);
17479: begin
17480:  TRGBType', 'record RedHex : string; GreenHex : string; BlueHex :'
17481:    + string; Red : integer; Green : integer; Blue : integer; end');
17482:  Function FadeColor( aColor : Longint; aFade : integer) : Tcolor');
17483:  Function CountColor( aColor : Tcolor; CompareColor : Tcolor; AdjustVale:integer):Tcolor;
17484: end;
17485:
17486: procedure SIRegister_uSettings(CL: TPSPascalCompiler);
17487: begin
17488:  Procedure SaveOscSettings');
17489:  Procedure GetOscSettings');
17490: end;
17491:
17492: procedure SIRegister_cyDebug(CL: TPSPascalCompiler);
17493: begin
17494:   TProcProcessEvent', 'Procedure ( Sender : TObject; Index : Integer)');
17495:   RecProcess', 'record Name : ShortString; DurationMs : Cardinal; '
17496:    FirstDurationMs : Cardinal; LastDurationMs : Cardinal; MinDurationMs : Int'
17497:    64; MaxDurationMs : Cardinal; MarksCount : Integer; ArrayMarks : array of '
17498:    Cardinal; EnterCount : Integer; ExitCount : Integer; end');
17499:   SIRegister_TcyDebug(CL);
17500: end;
17501:
17502: (*----------------------------------------------------------------------------*)
17503: procedure SIRegister_cyCopyFiles(CL: TPSPascalCompiler);
17504: begin
17505:   TCopyFileResult', '( cfCreate, cfOverwrite, cfNoNeed, cfForceDirectoryError, cfCopyError )');
17506:   TCopyFileExists', '( feDoNothing, feCopy, feCopyIfModified, feCopyIfMoreRecent )');
17507:   TCopyFileNotExists', '( fnDoNothing, fnCopy, fnCopyForceDir )');
17508:   SIRegister_TDestinationOptions(CL);
17509:   TProcCustomCopyFileEvent', 'Procedure ( Sender : TObject; var CopyFileResult : TCopyFileResult)');
17510:   TProcOnCopyFileProgressEvent','Procedure(Sender:TObject;FileBytes,
       TransferedBytes:int64;PercentDone:Int64);
17511:   TProcOnCustomSetFileDestination', 'Procedure ( Sender : TObject; var FileName : String)');
17512:   SIRegister_TcyCopyFiles(CL);
17513:  Function cyCopyFile(FromFile,ToFile: String; FileExists:TCopyFileExists;FileNotExists:TCopyFileNotExists;
       ResetAttr:boolean): TCopyFileResult');
17514:  Function cyCopyFileEx( FromFile,ToFile: String;FileExists: TCopyFileExists;FileNotExists
       TCopyFileNotExists; ResetAttr:boolean; aProgressBar:TProgressBar): TCopyFileResult');
17515:  Function cyCopyFilesEx(SourcePath, DestinationPath, IncludeFilters, ExcludeFilters : String; ArchiveFiles,
        ReadOnlyFiles, HiddenFiles, SystemFiles : TcyFileAttributeMode; FileExists : TCopyFileExists;'
17516:                      + FileNotExists: TCopyFileNotExists; SubFolders, ResetAttributes:Boolean) : Integer');
17517: end;
17518:
17519: procedure SIRegister_cySearchFiles(CL: TPSPascalCompiler);
17520: begin
17521:   CL.AddTypeS('TcyFileAttributeMode', '( faYes, faNo, faBoth )');
17522:   SIRegister_TcyFileAttributes(CL);
17523:   SIRegister_TSearchRecInstance(CL);
17524:   TOption', '( soOnlyDirs, soIgnoreAttributes, soIgnoreMaskInclude, soIgnoreMaskExclude )');
17525:   TOptions', 'set of TOption');
17526:   TSearchState', '( ssIdle, ssPaused, ssSearch, ssPausing, ssResuming, ssAborting )');
17527:   TProcOnValidateFileEvent Procedure(Sender:TObject;ValidMaskIncl,ValidMaskExcl,ValidAttribs:bool;var
       Accept:bool;
17528:   TProcOnValidateDirectoryEvent','Procedure(Sender:TObject;Directory:String;var Accept:boolean)');
17529:   SIRegister_TcyCustomSearchFiles(CL);
17530:   SIRegister_TcySearchFiles(CL);
```

```
17531:  Function FileNameRespondToMask( aFileName : String; aMask : String) : Boolean');
17532:  Function IscyFolder( aSRec : TSearchrec) : Boolean');
17533: end;
17534:
17535: procedure SIRegister_jcontrolutils(CL: TPSPascalCompiler);
17536: begin
17537:  Function jCountChar( const s : string; ch : char) : integer');
17538:  Procedure jSplit( const Delimiter : char; Input : string; Strings : TStrings)');
17539:  Function jNormalizeDate(const Value: string; theValue: TDateTime;const theFormat:string): string');
17540:  Function jNormalizeTime(const Value: string; theValue: TTime;const theFormat : string) : string');
17541:  Function jNormalizeDateTime(const Value:string;theValue:TDateTime;const theFormat:string):string');
17542:  Function jNormalizeDateSeparator( const s : string) : string');
17543:  Function jIsValidDateString( const Value : string) : boolean');
17544:  Function jIsValidTimeString( const Value : string) : boolean');
17545:  Function jIsValidDateTimeString( const Value : string) : boolean');
17546: end;
17547:
17548: procedure SIRegister_kcMapViewer(CL: TPSPascalCompiler);
17549: begin
17550:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMapViewer');
17551:   CL.AddTypeS('TMapSource', '( msNone, msGoogleNormal, msGoogleSatellite, msGoo'
17552:    +'gleHybrid, msGooglePhysical, msGooglePhysicalHybrid, msOpenStreetMapMapnik'
17553:    +', msOpenStreetMapOsmarender, msOpenCycleMap, msVirtualEarthBing, msVirtual'
17554:    +'EarthRoad, msVirtualEarthAerial, msVirtualEarthHybrid, msYahooNormal, msYa'
17555:    +'hooSatellite, msYahooHybrid, msOviNormal, msOviSatellite, msOviHybrid, msOviPhysical )');
17556:   TArea', 'record top : Int64; left : Int64; bottom : Int64; right: Int64; end');
17557:   TRealArea', 'record top : Extended; left : Extended; bottom : Extended; right : Extended; end');
17558:   TIntPoint', 'record X : Int64; Y : Int64; end');
17559:   TkcRealPoint', 'record X : Extended; Y : Extended; end');
17560:   TOnBeforeDownloadEvent', 'Procedure ( Url : string; str : TStream; var CanHandle : Boolean)');
17561:   TOnAfterDownloadEvent', 'Procedure ( Url : string; str : TStream)');
17562:   SIRegister_TCustomDownloadEngine(CL);
17563:   SIRegister_TCustomGeolocationEngine(CL);
17564:   SIRegister_TMapViewer(CL);
17565: end;
17566:
17567: procedure SIRegister_cparserutils(CL: TPSPascalCompiler);
17568: begin
17569:  (*CL.AddDelphiFunction('Function isFunc( name : TNamePart) : Boolean');
17570:  CL.AddDelphiFunction('Function isUnnamedFunc( name : TNamepart) : Boolean');
17571:  Function isPtrToFunc( name : TNamePart) : Boolean');
17572:  Function isFuncRetFuncPtr( name : TNamePart) : Boolean');
17573:  Function isPtrToFuncRetFuncPtr( name : TNamePart) : Boolean');
17574:  Function GetFuncParam( name : TNamePart) : TNamePart');
17575:  Function isArray( name : TNamePart) : Boolean');
17576:  Function GetArrayPart( name : TNamePart) : TNamePart');
17577:  Function GetIdFromPart( name : TNamePart) : AnsiString');
17578:  Function GetIdPart( name : TNamePart) : TNamePart');
17579:  Function isNamePartPtrToFunc( part : TNamePart) : Boolean');
17580:  Function isAnyBlock( part : TNamePart) : Boolean');*)
17581:   CL.AddTypeS('TLineInfo', 'record linestart : Integer; lineend : Integer; end');
17582:   SIRegister_TLineBreaker(CL);
17583:   CL.AddTypeS('TNameKind', 'Integer');
17584:   CL.AddClassN(CL.FindClass('TOBJECT'),'TNamePart');
17585:   //CL.AddTypeS('TFuncParam', 'record prmtype : TEntity; name : TNamePart; end');
17586:  Function SphericalMod( X : Extended) : Extended');
17587:  Function cSign( Value : Extended) : Extended');
17588:  Function LimitFloat( const eValue, eMin, eMax : Extended) : Extended');
17589:  Function AngleToRadians( iAngle : Extended) : Extended');
17590:  Function RadiansToAngle( eRad : Extended) : Extended');
17591:  Function Cross180( iLong : Double) : Boolean');
17592:  Function Mod180( Value : integer) : Integer');
17593:  Function Mod180Float( Value : Extended) : Extended');
17594:  Function MulDivFloat( a, b, d : Extended) : Extended');
17595:  Function LongDiff( iLong1, iLong2 : Double) : Double');
17596:  Procedure Bmp_AssignFromPersistent( Source : TPersistent; Bmp : TbitMap)');
17597:  Function Bmp_CreateFromPersistent( Source : TPersistent) : TbitMap');
17598:  Function FixFilePath( const Inpath, CheckPath : string) : string');
17599:  Function UnFixFilePath( const Inpath, CheckPath : string) : string');
17600:  Procedure FillStringList( sl : TStringList; const aText : string)');
17601: end;
17602:
17603: procedure SIRegister_LedNumber(CL: TPSPascalCompiler);
17604: begin
17605:   TLedSegmentSize', 'Integer');
17606:   TLedNumberBorderStyle', '( lnbNone, lnbSingle, lnbSunken, lnbRaised )');
17607:   SIRegister_TCustomLEDNumber(CL);
17608:   SIRegister_TLEDNumber(CL);
17609: end;
17610:
17611: procedure SIRegister_StStrL(CL: TPSPascalCompiler);
17612: begin
17613:   CL.AddTypeS('LStrRec', 'record AllocSize : Longint; RefCount : Longint; Length : Longint; end');
17614:    CL.AddTypeS('AnsiChar', 'Char');
17615:     CL.AddTypeS('BTable', 'array[0..255] of Byte');       //!!!
17616:   CL.AddDelphiFunction('Function HexBL( B : Byte) : AnsiString');
17617:  Function HexWL( W : Word) : AnsiString');
17618:  Function HexLL( L : LongInt) : AnsiString');
17619:  Function HexPtrL( P : ___Pointer) : AnsiString');
```

```
17620:  Function BinaryBL( B : Byte) : AnsiString');
17621:  Function BinaryWL( W : Word) : AnsiString');
17622:  Function BinaryLL( L : LongInt) : AnsiString');
17623:  Function OctalBL( B : Byte) : AnsiString');
17624:  Function OctalWL( W : Word) : AnsiString');
17625:  Function OctalLL( L : LongInt) : AnsiString');
17626:  Function Str2Int16L( const S : AnsiString; var I : SmallInt) : Boolean');
17627:  Function Str2WordL( const S : AnsiString; var I : Word) : Boolean');
17628:  Function Str2LongL( const S : AnsiString; var I : LongInt) : Boolean');
17629:  Function Str2RealL( const S : AnsiString; var R : Double) : Boolean');
17630:  Function Str2RealL( const S : AnsiString; var R : Real) : Boolean');
17631:  Function Str2ExtL( const S : AnsiString; var R : Extended) : Boolean');
17632:  Function Long2StrL( L : LongInt) : AnsiString');
17633:  Function Real2StrL( R : Double; Width : Byte; Places : ShortInt) : AnsiString');
17634:  Function Ext2StrL( R : Extended; Width : Byte; Places : ShortInt) : AnsiString');
17635:  Function ValPrepL( const S : AnsiString) : AnsiString');
17636:  Function CharStrL( C : Char; Len : Cardinal) : AnsiString');
17637:  Function PadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17638:  Function PadLL( const S : AnsiString; Len : Cardinal) : AnsiString');
17639:  Function LeftPadChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17640:  Function LeftPadL( const S : AnsiString; Len : Cardinal) : AnsiString');
17641:  Function TrimLeadL( const S : AnsiString) : AnsiString');
17642:  Function TrimTrailL( const S : AnsiString) : AnsiString');
17643:  Function TrimL( const S : AnsiString) : AnsiString');
17644:  Function TrimSpacesL( const S : AnsiString) : AnsiString');
17645:  Function CenterChL( const S : AnsiString; C : AnsiChar; Len : Cardinal) : AnsiString');
17646:  Function CenterL( const S : AnsiString; Len : Cardinal) : AnsiString');
17647:  Function EntabL( const S : AnsiString; TabSize : Byte) : AnsiString');
17648:  Function DetabL( const S : AnsiString; TabSize : Byte) : AnsiString');
17649:  Function ScrambleL( const S, Key : AnsiString) : AnsiString');
17650:  Function SubstituteL( const S, FromStr, ToStr : AnsiString) : AnsiString');
17651:  Function FilterL( const S, Filters : AnsiString) : AnsiString');
17652:  Function CharExistsL( const S : AnsiString; C : AnsiChar) : Boolean');
17653:  Function CharCountL( const S : AnsiString; C : AnsiChar) : Cardinal');
17654:  Function WordCountL( const S, WordDelims : AnsiString) : Cardinal');
17655:  Function WordPositionL( N : Cardinal; const S, WordDelims : AnsiString; var Pos : Cardinal) : Boolean');
17656:  Function ExtractWordL( N : Cardinal; const S, WordDelims : AnsiString) : AnsiString');
17657:  Function AsciiCountL( const S, WordDelims : AnsiString; Quote : AnsiChar) : Cardinal');
17658:  Function AsciiPositionL(N: Cardinal;const S,WordDelims:AnsiString;Quote:AnsiChar;var Pos:Cardinal):Bool;
17659:  Function ExtractAsciiL( N : Cardinal; const S, WordDelims : AnsiString; Quote : AnsiChar) : AnsiString');
17660:  Procedure WordWrapL(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17661:  Procedure WordWrap(const InSt:AnsiString;var OutSt,Overlap:AnsiString;Margin:Cardinal;PadToMargin:Bool;
17662:  Function CompStringL( const S1, S2 : AnsiString) : Integer');
17663:  Function CompUCStringL( const S1, S2 : AnsiString) : Integer');
17664:  Function SoundexL( const S : AnsiString) : AnsiString');
17665:  Function MakeLetterSetL( const S : AnsiString) : Longint');
17666:  Procedure BMMakeTableL( const MatchString : AnsiString; var BT : BTable)');
17667:  Function BMSearchL(var Buffer,BufLength:Cardinal;var BT:BTable;const MatchString:AnsiString;var
       Pos:Card):Bool;
17668:  Function BMSearchUCL( var Buffer, BufLength : Cardinal; var BT : BTable; const MatchString : AnsiString;
       var Pos : Cardinal) : Boolean');
17669:  Function DefaultExtensionL( const Name, Ext : AnsiString) : AnsiString');
17670:  Function ForceExtensionL( const Name, Ext : AnsiString) : AnsiString');
17671:  Function JustFilenameL( const PathName : AnsiString) : AnsiString');
17672:  Function JustNameL( const PathName : AnsiString) : AnsiString');
17673:  Function JustExtensionL( const Name : AnsiString) : AnsiString');
17674:  Function JustPathnameL( const PathName : AnsiString) : AnsiString');
17675:  Function AddBackSlashL( const DirName : AnsiString) : AnsiString');
17676:  Function CleanPathNameL( const PathName : AnsiString) : AnsiString');
17677:  Function HasExtensionL( const Name : AnsiString; var DotPos : Cardinal) : Boolean');
17678:  Function CommaizeL( L : LongInt) : AnsiString');
17679:  Function CommaizeChL( L : Longint; Ch : AnsiChar) : AnsiString');
17680:  Function FloatFormL(const Mask:AnsiString;R:TstFloat;const LtCurr,RtCurr:AnsiString;Sep,
       DecPt:Char):AnsiString;
17681:  Function LongIntFormL(const Mask:AnsiString;L:LongInt;const LtCurr,
       RtCurr:AnsiString;Sep:Char):AnsiString);
17682:  Function StrChPosL( const P : AnsiString; C : AnsiChar; var Pos : Cardinal) : Boolean');
17683:  Function StrStPosL( const P, S : AnsiString; var Pos : Cardinal) : Boolean');
17684:  Function StrStCopyL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17685:  Function StrChInsertL( const S : AnsiString; C : AnsiChar; Pos : Cardinal) : AnsiString');
17686:  Function StrStInsertL( const S1, S2 : AnsiString; Pos : Cardinal) : AnsiString');
17687:  Function StrChDeleteL( const S : AnsiString; Pos : Cardinal) : AnsiString');
17688:  Function StrStDeleteL( const S : AnsiString; Pos, Count : Cardinal) : AnsiString');
17689:  Function ContainsOnlyL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17690:  Function ContainsOtherThanL( const S, Chars : AnsiString; var BadPos : Cardinal) : Boolean');
17691:  Function CopyLeftL( const S : AnsiString; Len : Cardinal) : AnsiString');
17692:  Function CopyMidL( const S : AnsiString; First, Len : Cardinal) : AnsiString');
17693:  Function CopyRightL( const S : AnsiString; First : Cardinal) : AnsiString');
17694:  Function CopyRightAbsL( const S : AnsiString; NumChars : Cardinal) : AnsiString');
17695:  Function CopyFromNthWordL(const S,WordDelims:AString;const AWord:AString;N:Cardinal;var
       SubString:AString):Bool;
17696:  Function CopyFromToWordL(const S,WordDelims,Word1,Word2:AnsiString;N1,N2:Cardinal;var
       SubString:AnsiString):Bool;
17697:  Function CopyWithinL( const S, Delimiter : AnsiString; Strip : Boolean) : AnsiString');
17698:  Function DeleteFromNthWordL( const S, WordDelims : AnsiString; const AWord : AnsiString; N : Cardinal;
       var SubString : AnsiString) : Boolean');
17699:  Function DeleteFromToWordL( const S, WordDelims, Word1, Word2 : AnsiString; N1, N2 : Cardinal; var
       SubString : AnsiString) : Boolean');
17700:  Function DeleteWithinL( const S, Delimiter : AnsiString) : AnsiString');
```

```
17701:  Function ExtractTokensL(const S,
       Delims:AnsiString;QuoteChar:AnsiChar;AllowNulls:Bool;Tokens:TStrings):Cardinal;
17702:  Function IsChAlphaL( C : AnsiChar) : Boolean');
17703:  Function IsChNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17704:  Function IsChAlphaNumericL( C : AnsiChar; const Numbers : AnsiString) : Boolean');
17705:  Function IsStrAlphaL( const S : AnsiString) : Boolean');
17706:  Function IsStrNumericL( const S, Numbers : AnsiString) : Boolean');
17707:  Function IsStrAlphaNumericL( const S, Numbers : AnsiString) : Boolean');
17708:  Function KeepCharsL( const S, Chars : AnsiString) : AnsiString');
17709:  Function LastWordL( const S, WordDelims, AWord : AnsiString; var Position : Cardinal) : Boolean');
17710:  Function LastWordAbsL( const S, WordDelims : AnsiString; var Position : Cardinal) : Boolean');
17711:  Function LastStringL( const S, AString : AnsiString; var Position : Cardinal) : Boolean');
17712:  Function LeftTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17713:  Function ReplaceWordL(const S, WordDelims,OldWord,NewWord:AnsiString;N:Card;var
       Replacements:Card):AnsiString;
17714:  Function ReplaceWordAllL(const S,WordDelims,OldWord,NewWord:AnsiString;var
       Replacements:Cardinal):AnsiString');
17715:  Function ReplaceStringL(const S,OldString,NewString:AnsiString;N:Cardinal;var
       Replacements:Cardinal):AnsiString;
17716:  Function ReplaceStringAllL(const S,OldString,NewString:AnsiString; var Replacements:Cardinal):AnsiString;
17717:  Function RepeatStringL(const RepeatString:AnsiString;var Repetitions:Cardinal;MaxLen:Cardinal):AnsiString;
17718:  Function RightTrimCharsL( const S, Chars : AnsiString) : AnsiString');
17719:  Function StrWithinL( const S, SearchStr : string; Start : Cardinal; var Position : Cardinal) : boolean');
17720:  Function TrimCharsL( const S, Chars : AnsiString) : AnsiString');
17721:  Function WordPosL(const S,WordDelims,AWord: AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17722:  Function WordPos(const S,WordDelims,AWord:AnsiString;N:Cardinal;var Position:Cardinal):Bool;
17723:  end;
17724:
17725:
17726: {A simple Oscilloscope using  TWaveIn class.
17727:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
17728: uses
17729:   Forms,
17730:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
17731:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
17732:   uColorFunctions in 'uColorFunctions.pas',
17733:   AMixer in 'AMixer.pas',
17734:   uSettings in 'uSettings.pas',
17735:   UWavein4 in 'UWavein4.pas',
17736:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
17737:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
17738:
17739:
17740: Functions_max hex in the box maXbox
17741: functionslist.txt
17742: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98
17743:
17744: ********************************************************************************
17745: Procedure
17746: PROCEDURE SIZE 8289 8242 7507 7401 6792 6310 5971 4438 3797 3600
17747: Procedure ***************************Now the Procedure list******************
17748: Procedure ( ACol, ARow : Integer; Items : TStrings)
17749: Procedure ( Agg : TAggregate)
17750: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
17751: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
17752: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
17753: Procedure ( ASender : TObject; const ABytes : Integer)
17754: Procedure ( ASender : TObject; VStream : TStream)
17755: Procedure ( AThread : TIdThread)
17756: Procedure ( AWebModule : TComponent)
17757: Procedure ( Column : TColumn)
17758: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
17759: Procedure ( const iStart : integer; const sText : string)
17760: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
17761: Procedure ( Database : TDatabase; LoginParams : TStrings)
17762: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
17763: Procedure ( DATASET : TDATASET)
17764: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
17765: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
17766: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
17767: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
17768: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
17769: Procedure ( Done : Integer)
17770: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
17771: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
17772: Procedure ( HeaderControl:TCustomHeaderControl;Sect:THeaderSection;Width:Int;State:TSectionTrackState)
17773: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
17774: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
17775: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
17776: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
17777: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
17778: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
17779: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
17780: Procedure ( SENDER : TFIELD; const TEXT : String)
17781: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
17782: Procedure ( Sender : TIdTelnet; const Buffer : String)
17783: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
17784: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
```

```
17785: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17786: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
17787: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
17788: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
17789: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
17790: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
17791: Procedure ( Sender : TObject; Button : TMPBtnType)
17792: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
17793: Procedure ( Sender : TObject; Button : TUDBtnType)
17794: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
17795: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
17796: Procedure ( Sender : TObject; Column : TListColumn)
17797: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
17798: Procedure ( Sender : TObject; Connecting : Boolean)
17799: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
17800: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
17801: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
17802: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
17803: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
17804: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
17805: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
17806: Procedure ( Sender : TObject; Index : LongInt)
17807: Procedure ( Sender : TObject; Item : TListItem)
17808: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
17809: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
17810: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
17811: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
17812: Procedure ( Sender : TObject; Item : TListItem; var S : string)
17813: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
17814: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
17815: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
17816: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
17817: Procedure ( Sender : TObject; Node : TTreeNode)
17818: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
17819: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
17820: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
17821: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
17822: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
17823: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
17824: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
17825: Procedure ( Sender : TObject; Rect : TRect)
17826: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
17827: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
17828: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
17829: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
17830: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
17831: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
17832: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
17833: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
17834: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
17835: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
17836: Procedure ( Sender : TObject; Thread : TServerClientThread)
17837: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
17838: Procedure ( Sender : TObject; Username, Password : string)
17839: Procedure ( Sender : TObject; var AllowChange : Boolean)
17840: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
17841: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
17842: Procedure ( Sender : TObject; var Continue : Boolean)
17843: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
17844: Procedure ( Sender : TObject; var Username : string)
17845: Procedure ( Sender : TObject; Wnd : HWND)
17846: Procedure ( Sender : TToolbar; Button : TToolButton)
17847: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
17848: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
17849: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
17850: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
17851: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
17852: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
17853: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
17854: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
17855: procedure (Sender: TObject)
17856: procedure (Sender: TObject; var Done: Boolean)
17857: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
17858: procedure _T(Name: tbtString; v: Variant);
17859: Procedure AbandonSignalHandler( RtlSigNum : Integer)
17860: Procedure Abort
17861: Procedure About1Click( Sender : TObject)
17862: Procedure Accept( Socket : TSocket)
17863: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
17864: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
17865: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
17866: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
17867: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
17868: Procedure Add( Addend1, Addend2 : TMyBigInt)
17869: Procedure ADD( const AKEY, AVALUE : VARIANT)
17870: Procedure Add( const Key : string; Value : Integer)
17871: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
17872: Procedure ADD( FIELD : TFIELD)
```

```
17873: Procedure ADD( ITEM : TMENUITEM)
17874: Procedure ADD( POPUP : TPOPUPMENU)
17875: Procedure AddCharacters( xCharacters : TCharSet)
17876: Procedure AddDriver( const Name : string; List : TStrings)
17877: Procedure AddImages( Value : TCustomImageList)
17878: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
17879: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
17880: Procedure AddLoader( Loader : TBitmapLoader)
17881: Procedure ADDPARAM( VALUE : TPARAM)
17882: Procedure AddPassword( const Password : string)
17883: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
17884: Procedure AddState( oState : TniRegularExpressionState)
17885: Procedure AddStrings( Strings : TStrings);
17886: procedure AddStrings(Strings: TStrings);
17887: Procedure AddStrings1( Strings : TWideStrings);
17888: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
17889: Procedure AddToRecentDocs( const Filename : string)
17890: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
17891: Procedure AllFunctionsList1Click( Sender : TObject)
17892: procedure AllObjectsList1Click(Sender: TObject);
17893: Procedure Allocate( AAllocateBytes : Integer)
17894: procedure AllResourceList1Click(Sender: TObject);
17895: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
17896: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
17897: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
17898: Procedure AnsiFree( var s : AnsiString)
17899: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
17900: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
17901: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
17902: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
17903: Procedure AntiFreeze;
17904: Procedure APPEND
17905: Procedure Append( const S : WideString)
17906: procedure Append(S: string);
17907: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
17908: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
17909: Procedure AppendChunk( Val : OleVariant)
17910: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
17911: Procedure AppendStr( var Dest : string; S : string)
17912: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
17913: Procedure ApplyRange
17914: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17915: Procedure Arrange( Code : TListArrangement)
17916: procedure Assert(expr : Boolean; const msg: string);
17917: procedure Assert2(expr : Boolean; const msg: string);
17918: Procedure Assign( AList : TCustomBucketList)
17919: Procedure Assign( Other : TObject)
17920: Procedure Assign( Source : TDragObject)
17921: Procedure Assign( Source : TPersistent)
17922: Procedure Assign(Source: TPersistent)
17923: procedure Assign2(mystring, mypath: string);
17924: Procedure AssignCurValues( Source : TDataSet);
17925: Procedure AssignCurValues1( const CurValues : Variant);
17926: Procedure ASSIGNFIELD( FIELD : TFIELD)
17927: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
17928: Procedure AssignFile(var F: Text; FileName: string)
17929: procedure AssignFile(var F: TextFile; FileName: string)
17930: procedure AssignFileRead(var mystring, myfilename: string);
17931: procedure AssignFileWrite(mystring, myfilename: string);
17932: Procedure AssignTo( Other : TObject)
17933: Procedure AssignValues( Value : TParameters)
17934: Procedure ASSIGNVALUES( VALUE : TPARAMS)
17935: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
17936: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
17937: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
17938: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
17939: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17940: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
17941: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
17942: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
17943: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
17944: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
17945: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
17946: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
17947: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
17948: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17949: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
17950: procedure Beep
17951: Procedure BeepOk
17952: Procedure BeepQuestion
17953: Procedure BeepHand
17954: Procedure BeepExclamation
17955: Procedure BeepAsterisk
17956: Procedure BeepInformation
17957: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
17958: Procedure BeginLayout
17959: Procedure BeginTimer( const Delay, Resolution : Cardinal)
17960: Procedure BeginUpdate
17961: procedure BeginUpdate;
```

```
17962: procedure BigScreen1Click(Sender: TObject);
17963: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
17964: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
17965: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
17966: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
17967: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
17968: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
17969: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
17970: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
17971: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
17972: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
17973: Procedure BreakPointMenuClick( Sender : TObject)
17974: procedure BRINGTOFRONT
17975: procedure BringToFront;
17976: Procedure btnBackClick( Sender : TObject)
17977: Procedure btnBrowseClick( Sender : TObject)
17978: Procedure BtnClick( Index : TNavigateBtn)
17979: Procedure btnLargeIconsClick( Sender : TObject)
17980: Procedure BuildAndSendRequest( AURI : TIdURI)
17981: Procedure BuildCache
17982: Procedure BurnMemory( var Buff, BuffLen : integer)
17983: Procedure BurnMemoryStream( Destructo : TMemoryStream)
17984: Procedure CalculateFirstSet
17985: Procedure Cancel
17986: procedure CancelDrag;
17987: Procedure CancelEdit
17988: procedure CANCELHINT
17989: Procedure CancelRange
17990: Procedure CancelUpdates
17991: Procedure CancelWriteBuffer
17992: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
17993: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
17994: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
17995: procedure CaptureScreenFormat(vname: string; vextension: string);
17996: procedure CaptureScreenPNG(vname: string);
17997: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
17998: procedure CASCADE
17999: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
18000: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
18001: Procedure cbPathClick( Sender : TObject)
18002: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18003: Procedure cedebugAfterExecute( Sender : TPSScript)
18004: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18005: Procedure cedebugCompile( Sender : TPSScript)
18006: Procedure cedebugExecute( Sender : TPSScript)
18007: Procedure cedebugIdle( Sender : TObject)
18008: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
18009: Procedure CenterHeight( const pc, pcParent : TControl)
18010: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18011: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
18012: Procedure Change
18013: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
18014: Procedure Changed
18015: Procedure ChangeDir( const ADirName : string)
18016: Procedure ChangeDirUp
18017: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
18018: Procedure ChangeLevelBy( Value : TChangeRange)
18019: Procedure ChDir(const s: string)
18020: Procedure Check(Status: Integer)
18021: Procedure CheckCommonControl( CC : Integer)
18022: Procedure CHECKFIELDNAME( const FIELDNAME : String)
18023: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
18024: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
18025: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
18026: Procedure CheckToken( T : Char)
18027: procedure CheckToken(t:char)
18028: Procedure CheckTokenSymbol( const S : string)
18029: procedure CheckTokenSymbol(s:string)
18030: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
18031: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18032: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
18033: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
18034: procedure CipherFile1Click(Sender: TObject);
18035: Procedure Clear;
18036: Procedure Clear1Click( Sender : TObject)
18037: Procedure ClearColor( Color : TColor)
18038: Procedure CLEARITEM( AITEM : TMENUITEM)
18039: Procedure ClearMapping
18040: Procedure ClearSelection( KeepPrimary : Boolean)
18041: Procedure ClearWriteBuffer
18042: Procedure Click
18043: Procedure Close
18044: Procedure Close1Click( Sender : TObject)
18045: Procedure CloseDatabase( Database : TDatabase)
18046: Procedure CloseDataSets
18047: Procedure CloseDialog
18048: Procedure CloseFile(var F: Text);
18049: Procedure Closure
18050: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
```

```
18051: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18052: Procedure CodeCompletionList1Click( Sender : TObject)
18053: Procedure ColEnter
18054: Procedure Collapse
18055: Procedure Collapse( Recurse : Boolean)
18056: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
18057: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
18058: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
18059: Procedure Compile1Click( Sender : TObject)
18060: procedure ComponentCount1Click(Sender: TObject);
18061: Procedure Compress(azipfolder, azipfile: string)
18062: Procedure DeCompress(azipfolder, azipfile: string)
18063: Procedure XZip(azipfolder, azipfile: string)
18064: Procedure XUnZip(azipfolder, azipfile: string)
18065: Procedure Connect( const ATimeout : Integer)
18066: Procedure Connect( Socket : TSocket)
18067: procedure Console1Click(Sender: TObject);
18068: Procedure Continue
18069: Procedure ContinueCount( var Counter : TJclCounter)
18070: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
18071: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
18072: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
18073: Procedure ConvertImage(vsource, vdestination: string);
18074: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
18075: Procedure ConvertBitmap(vsource, vdestination: string);
18076: Procedure ConvertToGray(Cnv: TCanvas);
18077: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
18078: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
18079: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
18080: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
18081: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
18082: Procedure CopyFrom( mbCopy : TMyBigInt)
18083: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
18084: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
18085: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
18086: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
18087: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
18088: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
18089: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
18090: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
18091: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
18092: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18093: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
18094: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
18095: Procedure CopyToClipboard
18096: Procedure CountParts
18097: Procedure CreateDataSet
18098: Procedure CreateEmptyFile( const FileName : string)
18099: Procedure CreateFileFromString( const FileName, Data : string)
18100: Procedure CreateFromDelta( Source : TPacketDataSet)
18101: procedure CREATEHANDLE
18102: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
       OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
18103: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
18104: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
18105: Procedure CreateTable
18106: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
18107: procedure CSyntax1Click(Sender: TObject);
18108: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
18109: Procedure CURSORPOSCHANGED
18110: procedure CutFirstDirectory(var S: String)
18111: Procedure DataBaseError(const Message: string)
18112: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
18113: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
18114: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
18115: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
18116: Procedure DBIError(errorCode: Integer)
18117: Procedure DebugOutput( const AText : string)
18118: Procedure DebugRun1Click( Sender : TObject)
18119: procedure Dec;
18120: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
18121: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
18122: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
18123: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
18124: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
18125: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
18126: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
18127: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
18128: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
18129: Procedure Decompile1Click( Sender : TObject)
18130: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
18131: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
18132: Procedure DeferLayout
18133: Procedure defFileread
18134: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
18135: Procedure DelayMicroseconds( const MicroSeconds : Integer)
18136: Procedure Delete
```

```
18137: Procedure Delete( const AFilename : string)
18138: Procedure Delete( const Index : Integer)
18139: Procedure DELETE( INDEX : INTEGER)
18140: Procedure Delete( Index : LongInt)
18141: Procedure Delete( Node : TTreeNode)
18142: procedure Delete(var s: AnyString; ifrom, icount: Longint);
18143: Procedure DeleteAlias( const Name : string)
18144: Procedure DeleteDriver( const Name : string)
18145: Procedure DeleteIndex( const Name : string)
18146: Procedure DeleteKey( const Section, Ident : String)
18147: Procedure DeleteRecords
18148: Procedure DeleteRecords( AffectRecords : TAffectRecords)
18149: Procedure DeleteString( var pStr : String; const pDelStr : string)
18150: Procedure DeleteTable
18151: procedure DelphiSite1Click(Sender: TObject);
18152: Procedure Deselect
18153: Procedure Deselect( Node : TTreeNode)
18154: procedure DestroyComponents
18155: Procedure DestroyHandle
18156: Procedure Diff( var X : array of Double)
18157: procedure Diff(var X: array of Double);
18158: Procedure DirCreate( const DirectoryName : String)');
18159: procedure DISABLEALIGN
18160: Procedure DisableConstraints
18161: Procedure Disconnect
18162: Procedure Disconnect( Socket : TSocket)
18163: Procedure Dispose
18164: procedure Dispose(P: PChar)
18165: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
18166: Procedure DoKey( Key : TDBCtrlGridKey)
18167: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18168: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
18169: Procedure Dormant
18170: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
18171: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
18172: Procedure DoubleToComp( Value : Double; var Result : Comp)
18173: Procedure doWebCamPic(picname: string); //eg: c:\mypic.png
18174: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
18175: procedure Draw(X, Y: Integer; Graphic: TGraphic);
18176: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
18177: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18178: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
18179: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
18180: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
18181: procedure DrawFocusRect(const Rect: TRect);
18182: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
18183: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
18184: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
18185: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
18186: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
18187: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
18188: Procedure DropConnections
18189: Procedure DropDown
18190: Procedure DumpDescription( oStrings : TStrings)
18191: Procedure DumpStateTable( oStrings : TStrings)
18192: Procedure EDIT
18193: Procedure EditButtonClick
18194: Procedure EditFont1Click( Sender : TObject)
18195: procedure Ellipse(X1, Y1, X2, Y2: Integer);
18196: Procedure Ellipse1( const Rect : TRect);
18197: Procedure EMMS
18198: Procedure Encode( ADest : TStream)
18199: procedure ENDDRAG(DROP:BOOLEAN)
18200: Procedure EndEdit( Cancel : Boolean)
18201: Procedure EndTimer
18202: Procedure EndUpdate
18203: Procedure EraseSection( const Section : string)
18204: Procedure Error( const Ident : string)
18205: procedure Error(Ident:Integer)
18206: Procedure ErrorFmt( const Ident : string; const Args : array of const)
18207: Procedure ErrorStr( const Message : string)
18208: procedure ErrorStr(Message:String)
18209: Procedure Exchange( Index1, Index2 : Integer)
18210: procedure Exchange(Index1, Index2: Integer);
18211: Procedure Exec( FileName, Parameters, Directory : string)
18212: Procedure ExecProc
18213: Procedure ExecSQL( UpdateKind : TUpdateKind)
18214: Procedure Execute
18215: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
18216: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
18217: Procedure ExecuteCommand(executeFile, paramstring: string)
18218: Procedure ExecuteShell(executeFile, paramstring: string)
18219: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18220: Procedure ExitThread(ExitCode: Integer); stdcall;
18221: Procedure ExitProcess(ExitCode: Integer); stdcall;
18222: Procedure Expand( AUserName : String; AResults : TStrings)
18223: Procedure Expand( Recurse : Boolean)
```

```
18224: Procedure ExportClipboard1Click( Sender : TObject)
18225: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
18226: Procedure ExtractContentFields( Strings : TStrings)
18227: Procedure ExtractCookieFields( Strings : TStrings)
18228: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
18229: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
18230: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
18231: Procedure ExtractQueryFields( Strings : TStrings)
18232: Procedure FastDegToGrad
18233: Procedure FastDegToRad
18234: Procedure FastGradToDeg
18235: Procedure FastGradToRad
18236: Procedure FastRadToDeg
18237: Procedure FastRadToGrad
18238: Procedure FileClose( Handle : Integer)
18239: Procedure FileClose(handle: integer)
18240: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
18241: Procedure FileStructure( AStructure : TIdFTPDataStructure)
18242: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
18243: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
18244: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
18245: Procedure FillChar2(var X: PChar ; count: integer; value: char)
18246: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
18247: Procedure FillIPList
18248: procedure FillRect(const Rect: TRect);
18249: Procedure FillTStrings( AStrings : TStrings)
18250: Procedure FilterOnBookmarks( Bookmarks : array of const)
18251: procedure FinalizePackage(Module: HMODULE)
18252: procedure FindClose;
18253: procedure FindClose2(var F: TSearchRec)
18254: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
18255: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
18256: Procedure FindNearest( const KeyValues : array of const)
18257: Procedure FinishContext
18258: Procedure FIRST
18259: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
18260: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
18261: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
18262: Procedure FlushSchemaCache( const TableName : string)
18263: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
18264: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
18265: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
18266: Procedure FormActivate( Sender : TObject)
18267: procedure FormatLn(const format: String; const args: array of const);  //alias
18268: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18269: Procedure FormCreate( Sender : TObject)
18270: Procedure FormDestroy( Sender : TObject)
18271: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18272: procedure FormOutput1Click(Sender: TObject);
18273: Procedure FormToHtml( Form : TForm; Path : string)
18274: procedure FrameRect(const Rect: TRect);
18275: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
18276: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
18277: Procedure Free( Buffer : TRecordBuffer)
18278: Procedure Free( Buffer : TValueBuffer)
18279: Procedure Free;
18280: Procedure FreeAndNil(var Obj:TObject)
18281: Procedure FreeImage
18282: procedure FreeMem(P: PChar; Size: Integer)
18283: Procedure FreeTreeData( Tree : TUpdateTree)
18284: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
18285: Procedure FullCollapse
18286: Procedure FullExpand
18287: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
18288: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
18289: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
18290: Procedure Get1( AURL : string; const AResponseContent : TStream);
18291: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
18292: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
18293: Procedure GetAliasNames( List : TStrings)
18294: Procedure GetAliasParams( const AliasName : string; List : TStrings)
18295: Procedure GetApplicationsRunning( Strings : TStrings)
18296: Procedure getBox(aURL, extension: string);
18297: Procedure GetCommandTypes( List : TWideStrings)
18298: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
18299: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
18300: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
18301: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
18302: Procedure GetDatabaseNames( List : TStrings)
18303: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
18304: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
18305: Procedure GetDir(d: byte; var s: string)
18306: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
18307: Procedure GetDriverNames( List : TStrings)
18308: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
18309: Procedure GetDriverParams( const DriverName : string; List : TStrings)
18310: Procedure GetEMails1Click( Sender : TObject)
```

```
18311: Procedure getEnvironmentInfo;
18312: Function getEnvironmentString: string;
18313: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
18314: Procedure GetFieldNames( const TableName : string; List : TStrings)
18315: Procedure GetFieldNames( const TableName : string; List : TStrings);
18316: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
18317: Procedure GETFIELDNAMES( LIST : TSTRINGS)
18318: Procedure GetFieldNames1( const TableName : string; List : TStrings);
18319: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
18320: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
18321: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
18322: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
18323: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
18324: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
18325: Procedure GetFormatSettings
18326: Procedure GetFromDIB( var DIB : TBitmapInfo)
18327: Procedure GetFromHDIB( HDIB : HBitmap)
18328: // GetGEOMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne')
18329: Procedure GetGEOMap(C_form,apath: string; const Data: string); //c_form: [html/json/xml]
18330: Procedure GetIcon( Index : Integer; Image : TIcon);
18331: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
18332: Procedure GetIndexInfo( IndexName : string)
18333: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
18334: Procedure GetIndexNames( List : TStrings)
18335: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
18336: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
18337: Procedure GetIndexNames4( const TableName : string; List : TStrings);
18338: Procedure GetInternalResponse
18339: Procedure GETITEMNAMES( LIST : TSTRINGS)
18340: procedure GetMem(P: PChar; Size: Integer)
18341: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
18342: procedure GetPackageDescription(ModuleName: PChar): string)
18343: Procedure GetPackageNames( List : TStrings);
18344: Procedure GetPackageNames1( List : TWideStrings);
18345: Procedure GetParamList( List : TList; const ParamNames : WideString)
18346: Procedure GetProcedureNames( List : TStrings);
18347: Procedure GetProcedureNames( List : TWideStrings);
18348: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
18349: Procedure GetProcedureNames1( List : TStrings);
18350: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
18351: Procedure GetProcedureNames3( List : TWideStrings);
18352: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
18353: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
18354: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
18355: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
18356: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
18357: Procedure GetProviderNames( Names : TWideStrings);
18358: Procedure GetProviderNames( Proc : TGetStrProc)
18359: Procedure GetProviderNames1( Names : TStrings);
18360: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
18361: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no open image
18362: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
18363: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
18364: Procedure GetSchemaNames( List : TStrings);
18365: Procedure GetSchemaNames1( List : TWideStrings);
18366: Procedure getScriptandRunAsk;
18367: Procedure getScriptandRun(ascript: string);
18368: Procedure getScript(ascript: string);   //alias
18369: Procedure getWebScript(ascript: string);   //alias
18370: Procedure GetSessionNames( List : TStrings)
18371: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
18372: Procedure GetStrings( List : TStrings)
18373: Procedure GetSystemTime; stdcall;
18374: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
18375: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
18376: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
18377: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
18378: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
18379: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
18380: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
18381: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
18382: Procedure GetVisibleWindows( List : Tstrings)
18383: Procedure GoBegin
18384: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
18385: Procedure GotoCurrent( Table : TTable)
18386: procedure GotoEnd1Click(Sender: TObject);
18387: Procedure GotoNearest
18388: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
18389: Procedure HandleException( E : Exception; var Handled : Boolean)
18390: procedure HANDLEMESSAGE
18391: procedure HandleNeeded;
18392: Procedure Head( AURL : string)
18393: Procedure Help( var AHelpContents : TStringList; ACommand : String)
18394: Procedure HexToBinary( Stream : TStream)
18395: procedure HexToBinary(Stream:TStream)
18396: Procedure HideDragImage
18397: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
```

```
18398: Procedure HideTraybar
18399: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
18400: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
18401: Procedure HookOSExceptions
18402: Procedure HookSignal( RtlSigNum : Integer)
18403: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
18404: Procedure HTMLSyntax1Click( Sender : TObject)
18405: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
18406: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
18407: Procedure ImportfromClipboard1Click( Sender : TObject)
18408: Procedure ImportfromClipboard2Click( Sender : TObject)
18409: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
18410: procedure Incb(var x: byte);
18411: Procedure Include1Click( Sender : TObject)
18412: Procedure IncludeOFF;  //preprocessing
18413: Procedure IncludeON;
18414: procedure Info1Click(Sender: TObject);
18415: Procedure InitAltRecBuffers( CheckModified : Boolean)
18416: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
18417: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
18418: Procedure InitData( ASource : TDataSet)
18419: Procedure InitDelta( ADelta : TPacketDataSet);
18420: Procedure InitDelta1( const ADelta : OleVariant);
18421: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
18422: Procedure Initialize
18423: procedure InitializePackage(Module: HMODULE)
18424: Procedure INITIATEACTION
18425: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
18426: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
18427: Procedure InitModule( AModule : TComponent)
18428: Procedure InitStdConvs
18429: Procedure InitTreeData( Tree : TUpdateTree)
18430: Procedure INSERT
18431: Procedure Insert( Index : Integer; AClass : TClass)
18432: Procedure Insert( Index : Integer; AComponent : TComponent)
18433: Procedure Insert( Index : Integer; AObject : TObject)
18434: Procedure Insert( Index : Integer; const S : WideString)
18435: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
18436: Procedure Insert(Index: Integer; const S: string);
18437: procedure Insert(Index: Integer; S: string);
18438: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
18439: procedure InsertComponent(AComponent:TComponent)
18440: procedure InsertControl(AControl: TControl);
18441: Procedure InsertIcon( Index : Integer; Image : TIcon)
18442: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
18443: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
18444: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
18445: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
18446: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
18447: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
18448: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
18449: Procedure InternalBeforeResolve( Tree : TUpdateTree)
18450: Procedure InvalidateModuleCache
18451: Procedure InvalidateTitles
18452: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
18453: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
18454: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
18455: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
18456: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
18457: procedure JavaSyntax1Click(Sender: TObject);
18458: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
18459: Procedure KillDataChannel
18460: Procedure Largefont1Click( Sender : TObject)
18461: Procedure LAST
18462: Procedure LaunchCpl( FileName : string)
18463: Procedure Launch( const AFile : string)
18464: Procedure LaunchFile( const AFile : string)
18465: Procedure LetFileList(FileList: TStringlist; apath: string);
18466: Procedure lineToNumber( xmemo : String; met : boolean)
18467: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
18468: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
18469: Procedure ListViewData( Sender : TObject; Item : TListItem)
18470: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
18471: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
18472: Procedure ListViewDblClick( Sender : TObject)
18473: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
18474: Procedure ListDLLExports(const FileName: string; List: TStrings);
18475: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
18476: procedure LoadBytecode1Click(Sender: TObject);
18477: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
18478: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
18479: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
18480: Procedure LoadFromFile( AFileName : string)
18481: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
18482: Procedure LoadFromFile( const FileName : string)
```

```
18483: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
18484: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
18485: Procedure LoadFromFile( const FileName : WideString)
18486: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18487: Procedure LoadFromFile(const AFileName: string)
18488: procedure LoadFromFile(FileName: string);
18489: procedure LoadFromFile(FileName:String)
18490: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
18491: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
18492: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
18493: Procedure LoadFromStream( const Stream : TStream)
18494: Procedure LoadFromStream( S : TStream)
18495: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18496: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18497: Procedure LoadFromStream( Stream : TStream)
18498: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
18499: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
18500: procedure LoadFromStream(Stream: TStream);
18501: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
18502: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
18503: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
18504: Procedure LoadLastFile1Click( Sender : TObject)
18505: { LoadIcoToImage loads two icons from resource named NameRes,
18506:   into two image lists ALarge and ASmall}
18507: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
18508: Procedure LoadMemo
18509: Procedure LoadParamsFromIniFile( FFileName : WideString)
18510: Procedure Lock
18511: Procedure Login
18512: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
18513: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
18514: Procedure MakeCaseInsensitive
18515: Procedure MakeDeterministic( var bChanged : boolean)
18516: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
18517: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
18518: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
18519: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
18520: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
18521: Procedure SetRectComplexFormatStr( const S : string)
18522: Procedure SetPolarComplexFormatStr( const S : string)
18523: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
18524: Procedure MakeVisible
18525: Procedure MakeVisible( PartialOK : Boolean)
18526: Procedure Manual1Click( Sender : TObject)
18527: Procedure MarkReachable
18528: Procedure maXbox;  //shows the exe version data in a win box
18529: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
18530: Procedure Memo1Change( Sender : TObject)
18531: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
18532: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
18533: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18534: procedure Memory1Click(Sender: TObject);
18535: Procedure MERGE( MENU : TMAINMENU)
18536: Procedure MergeChangeLog
18537: procedure MINIMIZE
18538: Procedure MinimizeMaxbox;
18539: Procedure MkDir(const s: string)
18540: Procedure MakeDir(const s: string)');
18541: Procedure ChangeDir(const s: string)');
18542: Function makeFile(const FileName: string): integer)');
18543: Procedure mnuPrintFont1Click( Sender : TObject)
18544: procedure ModalStarted
18545: Procedure Modified
18546: Procedure ModifyAlias( Name : string; List : TStrings)
18547: Procedure ModifyDriver( Name : string; List : TStrings)
18548: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
18549: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
18550: Procedure Move( CurIndex, NewIndex : Integer)
18551: procedure Move(CurIndex, NewIndex: Integer);
18552: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
18553: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
18554: Procedure moveCube( o : TMyLabel)
18555: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
18556: procedure MoveTo(X, Y: Integer);
18557: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
18558: Procedure MovePoint(var x,y:Extended; const angle:Extended);
18559: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
18560: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
18561: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
18562: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
18563: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
18564: procedure New(P: PChar)
18565: procedure New1Click(Sender: TObject);
18566: procedure NewInstance1Click(Sender: TObject);
18567: Procedure NEXT
18568: Procedure NextMonth
18569: Procedure Noop
18570: Procedure NormalizePath( var APath : string)
```

```
18571: procedure ObjectBinaryToText(Input, Output: TStream)
18572: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18573: procedure ObjectResourceToText(Input, Output: TStream)
18574: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18575: procedure ObjectTextToBinary(Input, Output: TStream)
18576: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18577: procedure ObjectTextToResource(Input, Output: TStream)
18578: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
18579: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
18580: Procedure Open( const UserID : WideString; const Password : WideString);
18581: Procedure Open;
18582: Procedure open1Click( Sender : TObject)
18583: Procedure OpenCdDrive
18584: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
18585: Procedure OpenCurrent
18586: Procedure OpenFile(vfilenamepath: string)
18587: Procedure OpenDirectory1Click( Sender : TObject)
18588: Procedure OpenDir(adir: string);
18589: Procedure OpenIndexFile( const IndexName : string)
18590: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
18591: Procedure OpenWriteBuffer( const AThreshhold : Integer)
18592: Procedure OptimizeMem
18593: Procedure Options1( AURL : string);
18594: Procedure OutputDebugString(lpOutputString : PChar)
18595: Procedure PackBuffer
18596: Procedure Paint
18597: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
18598: Procedure PaintToTBitmap( Target : TBitmap)
18599: Procedure PaletteChanged
18600: Procedure ParentBiDiModeChanged
18601: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
18602: Procedure PasteFromClipboard;
18603: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
18604: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
18605: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
18606: Procedure PError( Text : string)
18607: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
18608: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
18609: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
18610: procedure playmp3(mpath: string);
18611: Procedure PlayMP31Click( Sender : TObject)
18612: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
18613: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
18614: procedure PolyBezier(const Points: array of TPoint);
18615: procedure PolyBezierTo(const Points: array of TPoint);
18616: procedure Polygon(const Points: array of TPoint);
18617: procedure Polyline(const Points: array of TPoint);
18618: Procedure Pop
18619: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
18620: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
18621: Procedure POPUP( X, Y : INTEGER)
18622: Procedure PopupURL(URL : WideString);
18623: Procedure POST
18624: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
18625: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
18626: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
18627: Procedure PostUser( const Email, FirstName, LastName : WideString)
18628: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18629: procedure Pred(X: int64);
18630: Procedure Prepare
18631: Procedure PrepareStatement
18632: Procedure PreProcessXML( AList : TStrings)
18633: Procedure PreventDestruction
18634: Procedure Print( const Caption : string)
18635: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
18636: procedure printf(const format: String; const args: array of const);
18637: Procedure PrintList(Value: TStringList);
18638: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
18639: Procedure Printout1Click( Sender : TObject)
18640: Procedure ProcessHeaders
18641: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
18642: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
18643: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
18644: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
18645: Procedure ProcessMessagesOFF;  //application.processmessages
18646: Procedure ProcessMessagesON;
18647: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
18648: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
18649: Procedure Proclist Size is: 3797 /1415
18650: Procedure procMessClick( Sender : TObject)
18651: Procedure PSScriptCompile( Sender : TPSScript)
18652: Procedure PSScriptExecute( Sender : TPSScript)
18653: Procedure PSScriptLine( Sender : TObject)
18654: Procedure Push( ABoundary : string)
18655: procedure PushItem(AItem: Pointer)
18656: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
18657: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
18658: procedure PutLinuxLines(const Value: string)
```

```
18659: Procedure Quit
18660: Procedure RaiseConversionError( const AText : string);
18661: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
18662: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
18663: procedure RaiseException(Ex: TIFException; Param: String);
18664: Procedure RaiseExceptionForLastCmdResult;
18665: procedure RaiseLastException;
18666: procedure RaiseException2;
18667: Procedure RaiseException3(const Msg: string);
18668: Procedure RaiseExcept(const Msg: string);
18669: Procedure RaiseLastOSError
18670: Procedure RaiseLastWin32;
18671: procedure RaiseLastWin32Error)
18672: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
18673: Procedure RandomFillStream( Stream : TMemoryStream)
18674: procedure randomize;
18675: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
18676: Procedure RCS
18677: Procedure Read( Socket : TSocket)
18678: Procedure ReadBlobData
18679: procedure ReadBuffer(Buffer:String;Count:LongInt)
18680: procedure ReadOnly1Click(Sender: TObject);  -->maxform1.memo2.readonly:= false;
18681: Procedure ReadSection( const Section : string; Strings : TStrings)
18682: Procedure ReadSections( Strings : TStrings)
18683: Procedure ReadSections( Strings : TStrings);
18684: Procedure ReadSections1( const Section : string; Strings : TStrings);
18685: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
18686: Procedure ReadStream( AStream : TStream; ABytecount:LongInt; const AReadUntilDisconnect : boolean)
18687: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
18688: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
18689: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
18690: Procedure Realign;
18691: procedure Rectangle(X1, Y1, X2, Y2: Integer);
18692: Procedure Rectangle1( const Rect : TRect);
18693: Procedure RectCopy( var Dest : TRect; const Source : TRect)
18694: Procedure RectFitToScreen( var R : TRect)
18695: Procedure RectGrow( var R : TRect; const Delta : Integer)
18696: Procedure RectGrowX( var R : TRect; const Delta : Integer)
18697: Procedure RectGrowY( var R : TRect; const Delta : Integer)
18698: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
18699: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
18700: Procedure RectNormalize( var R : TRect)
18701: //  TFileCallbackProcedure = procedure(filename:string);
18702: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
18703: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
18704: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
18705: Procedure Refresh;
18706: Procedure RefreshData( Options : TFetchOptions)
18707: Procedure REFRESHLOOKUPLIST
18708: Procedure regExPathfinder(Pathin, fileout, firstp, aregex, ext: string; asort: boolean);
18709: Procedure RegExPathfinder2(Pathin, fileout, firstp, aregex, ext: string; asort, acopy: boolean);
18710: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
18711: Procedure RegisterChanges( Value : TChangeLink)
18712: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
18713: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
18714: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
18715: Procedure ReInitialize( ADelay : Cardinal)
18716: procedure RELEASE
18717: Procedure Remove( const ABytecount : integer)
18718: Procedure REMOVE( FIELD : TFIELD)
18719: Procedure REMOVE( ITEM : TMENUITEM)
18720: Procedure REMOVE( POPUP : TPOPUPMENU)
18721: Procedure RemoveAllPasswords
18722: procedure RemoveComponent(AComponent:TComponent)
18723: Procedure RemoveDir( const ADirName : string)
18724: Procedure RemoveLambdaTransitions( var bChanged : boolean)
18725: Procedure REMOVEPARAM( VALUE : TPARAM)
18726: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
18727: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
18728: Procedure Rename( const ASourceFile, ADestFile : string)
18729: Procedure Rename( const FileName : string; Reload : Boolean)
18730: Procedure RenameTable( const NewTableName : string)
18731: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
18732: Procedure Replace1Click( Sender : TObject)
18733: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
18734: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
18735: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
18736: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
18737: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
18738: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
18739: Procedure Requery( Options : TExecuteOptions)
18740: Procedure Reset
18741: Procedure Reset1Click( Sender : TObject)
18742: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
18743: procedure ResourceExplore1Click(Sender: TObject);
18744: Procedure RestoreContents
18745: Procedure RestoreDefaults
18746: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
18747: Procedure RetrieveHeaders
```

```
18748: Procedure RevertRecord
18749: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
18750: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18751: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18752: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
18753: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
18754: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
18755: Procedure RleCompress2( Stream : TStream)
18756: Procedure RleDecompress2( Stream : TStream)
18757: Procedure RmDir(const S: string)
18758: Procedure Rollback
18759: Procedure Rollback( TransDesc : TTransactionDesc)
18760: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
18761: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
18762: Procedure RollbackTrans
18763: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
18764: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
18765: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
18766: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
18767: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
18768: Procedure S_EBox( const AText : string)
18769: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
18770: Procedure S_IBox( const AText : string)
18771: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
18772: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
18773: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
18774: Procedure SampleVarianceAndMean
18775: ( const X : TDynFloatArray; var Variance, Mean : Float)
18776: Procedure Save2Click( Sender : TObject)
18777: Procedure Saveas3Click( Sender : TObject)
18778: Procedure Savebefore1Click( Sender : TObject)
18779: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
18780: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
18781: Procedure SaveConfigFile
18782: Procedure SaveOutput1Click( Sender : TObject)
18783: procedure SaveScreenshotClick(Sender: TObject);
18784: Procedure SaveLn(pathname, content: string);   //SaveIn(exepath+'mysavelntest.txt', memo2.text);
18785: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
18786: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
18787: Procedure SaveToFile( AFileName : string)
18788: Procedure SAVETOFILE( const FILENAME : String)
18789: Procedure SaveToFile( const FileName : WideString)
18790: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
18791: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18792: procedure SaveToFile(FileName: string);
18793: procedure SaveToFile(FileName:String)
18794: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
18795: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
18796: Procedure SaveToStream( S : TStream)
18797: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18798: Procedure SaveToStream( Stream : TStream)
18799: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
18800: procedure SaveToStream(Stream: TStream);
18801: procedure SaveToStream(Stream:TStream)
18802: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
18803: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
18804: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
18805: procedure Say(const sText: string)
18806: Procedure SBytecode1Click( Sender : TObject)
18807: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
18808: procedure ScriptExplorer1Click(Sender: TObject);
18809: Procedure Scroll( Distance : Integer)
18810: Procedure Scroll( DX, DY : Integer)
18811: procedure ScrollBy(DeltaX, DeltaY: Integer);
18812: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
18813: Procedure ScrollTabs( Delta : Integer)
18814: Procedure Search1Click( Sender : TObject)
18815: procedure SearchAndOpenDoc(vfilenamepath: string)
18816: procedure SearchAndOpenFile(vfilenamepath: string)
18817: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
18818: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
18819: Procedure SearchNext1Click( Sender : TObject)
18820: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
18821: Procedure Select1( const Nodes : array of TTreeNode);
18822: Procedure Select2( Nodes : TList);
18823: Procedure SelectNext( Direction : Boolean)
18824: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
18825: Procedure SelfTestPEM  //unit uPSI_cPEM
18826: Procedure Send( AMsg : TIdMessage)
18827: //config forst in const MAILINIFILE = 'maildef.ini';
18828: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
18829: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18830: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18831: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
18832: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
18833: Procedure SendResponse
18834: Procedure SendStream( AStream : TStream)
18835: Procedure Set8087CW( NewCW : Word)
```

```
18836: Procedure SetAll( One, Two, Three, Four : Byte)
18837: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
18838: Procedure SetAppDispatcher( const ADispatcher : TComponent)
18839: procedure SetArrayLength;
18840: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
18841: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
18842: procedure SetArrayLength2String2(arr: T2StringArray; asize1, asize2: integer);  //all init
18843: procedure SetArrayLength2Integer2(arr: T2IntegerArray; asize1, asize2: integer);
18844: procedure Set2DimStrArray(var arr: T2StringArray; asize1, asize2: integer);');
18845: procedure Set2DimIntArray(var arr: T2IntegerArray; asize1, asize2: integer);');
18846: procedure Set3DimIntArray(var arr: T3IntegerArray; asize1, asize2, asize3: integer);
18847: procedure Set3DimStrArray(var arr: T3StringArray; asize1, asize2, asize3: integer);
18848: Procedure SetAsHandle( Format : Word; Value : THandle)
18849: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
18850: procedure SetCaptureControl(Control: TControl);
18851: Procedure SetColumnAttributes
18852: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
18853: Procedure SetCustomHeader( const Name, Value : string)
18854: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
18855: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
18856: Procedure SetFocus
18857: procedure SetFocus; virtual;
18858: Procedure SetInitialState
18859: Procedure SetKey
18860: procedure SetLastError(ErrorCode: Integer)
18861: procedure SetLength;
18862: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
18863: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
18864: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
18865: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
18866: Procedure SetParams1( UpdateKind : TUpdateKind);
18867: Procedure SetPassword( const Password : string)
18868: Procedure SetPointer( Ptr : Pointer; Size : Longint)
18869: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
18870: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
18871: Procedure SetProvider( Provider : TComponent)
18872: Procedure SetProxy( const Proxy : string)
18873: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
18874: Procedure SetRange( const StartValues, EndValues : array of const)
18875: Procedure SetRangeEnd
18876: Procedure SetRate( const aPercent, aYear : integer)
18877: procedure SetRate(const aPercent, aYear: integer)
18878: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
18879: Procedure SetSafeCallExceptionMsg( Msg : String)
18880: procedure SETSELTEXTBUF(BUFFER:PCHAR)
18881: Procedure SetSize( AWidth, AHeight : Integer)
18882: procedure SetSize(NewSize:LongInt)
18883: procedure SetString(var s: string; buffer: PChar; len: Integer)
18884: Procedure SetStrings( List : TStrings)
18885: Procedure SetText( Text : PwideChar)
18886: procedure SetText(Text: PChar);
18887: Procedure SetTextBuf( Buffer : PChar)
18888: procedure SETTEXTBUF(BUFFER:PCHAR)
18889: Procedure SetTick( Value : Integer)
18890: Procedure SetTimeout( ATimeOut : Integer)
18891: Procedure SetTraceEvent( Event : TDBXTraceEvent)
18892: Procedure SetUserName( const UserName : string)
18893: Procedure SetWallpaper( Path : string);
18894: procedure ShellStyle1Click(Sender: TObject);
18895: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
18896: Procedure ShowFileProperties( const FileName : string)
18897: Procedure ShowInclude1Click( Sender : TObject)
18898: Procedure ShowInterfaces1Click( Sender : TObject)
18899: Procedure ShowLastException1Click( Sender : TObject)
18900: Procedure ShowMessage( const Msg : string)
18901: Procedure ShowMessageBig(const aText : string);
18902: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
18903: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
18904: Procedure MsgBig(const aText : string);        //alias
18905: procedure showmessage(mytext: string);
18906: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
18907: procedure ShowMessageFmt(const Msg: string; Params: array of const))
18908: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
18909: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
18910: Procedure ShowSearchDialog( const Directory : string)
18911: Procedure ShowSpecChars1Click( Sender : TObject)
18912: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
18913: Procedure ShredFile( const FileName : string; Times : Integer)
18914: procedure Shuffle(vQ: TStringList);
18915: Procedure ShuffleList( var List : array of Integer; Count : Integer)
18916: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
18917: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
18918: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
18919: Procedure Site( const ACommand : string)
18920: Procedure SkipEOL
18921: Procedure Sleep( ATime : cardinal)
18922: Procedure Sleep( milliseconds : Cardinal)
18923: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
```

```
18924: Procedure Slinenumbers1Click( Sender : TObject)
18925: Procedure Sort
18926: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
18927: procedure Speak(const sText: string)   //async like voice
18928: procedure Speak2(const sText: string)  //sync
18929: procedure Split(Str: string;  SubStr: string; List: TStrings);
18930: Procedure SplitNameValue( const Line : string; var Name, Value : string)
18931: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
18932: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
18933: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
18934: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
18935: procedure SQLSyntax1Click(Sender: TObject);
18936: Procedure SRand( Seed : RNG_IntType)
18937: Procedure Start
18938: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
18939: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
18940: //Ex.  StartFileFinder3(exepath+'exercices','*.pas','record',false,seclist);
18941: Procedure StartTransaction( TransDesc : TTransactionDesc)
18942: Procedure Status( var AStatusList : TStringList)
18943: Procedure StatusBar1DblClick( Sender : TObject)
18944: Procedure StepInto1Click( Sender : TObject)
18945: Procedure StepIt
18946: Procedure StepOut1Click( Sender : TObject)
18947: Procedure Stop
18948: procedure stopmp3;
18949: procedure StartWeb(aurl: string);
18950: Procedure Str(aint: integer; astr: string); //of system
18951: Procedure StrDispose( Str : PChar)
18952: procedure StrDispose(Str: PChar)
18953: Procedure StrReplace(var Str: String; Old, New: String);
18954: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
18955: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
18956: Procedure StringToBytes( Value : String; Bytes : array of byte)
18957: procedure StrSet(c : Char; I : Integer; var s : String);
18958: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
18959: Procedure StructureMount( APath : String)
18960: procedure STYLECHANGED(SENDER:TOBJECT)
18961: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
18962: procedure Succ(X: int64);
18963: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
18964: procedure SwapChar(var X,Y: char);  //swapX follows
18965: Procedure SwapFloats( var X, Y : Float)
18966: procedure SwapGrid(grd: TStringGrid);
18967: Procedure SwapOrd( var I, J : Byte);
18968: Procedure SwapOrd( var X, Y : Integer)
18969: Procedure SwapOrd1( var I, J : Shortint);
18970: Procedure SwapOrd2( var I, J : Smallint);
18971: Procedure SwapOrd3( var I, J : Word);
18972: Procedure SwapOrd4( var I, J : Integer);
18973: Procedure SwapOrd5( var I, J : Cardinal);
18974: Procedure SwapOrd6( var I, J : Int64);
18975: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
18976: Procedure Synchronize1( Method : TMethod);
18977: procedure SyntaxCheck1Click(Sender: TObject);
18978: Procedure SysFreeString(const S: WideString); stdcall;
18979: Procedure TakeOver( Other : TLinearBitmap)
18980: Procedure Talkln(const sText: string)  //async voice
18981: procedure tbtn6resClick(Sender: TObject);
18982: Procedure tbtnUseCaseClick( Sender : TObject)
18983: procedure TerminalStyle1Click(Sender: TObject);
18984: Procedure Terminate
18985: Procedure texSyntax1Click( Sender : TObject)
18986: procedure TextOut(X, Y: Integer; Text: string);
18987: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
18988: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
18989: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
18990: Procedure TextStart
18991: procedure TILE
18992: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
18993: Procedure TitleClick( Column : TColumn)
18994: Procedure ToDo
18995: procedure toolbtnTutorialClick(Sender: TObject);
18996: Procedure Trace1( AURL : string; const AResponseContent : TStream);
18997: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
18998: Procedure Truncate
18999: procedure Tutorial101Click(Sender: TObject);
19000: procedure Tutorial10Statistics1Click(Sender: TObject);
19001: procedure Tutorial11Forms1Click(Sender: TObject);
19002: procedure Tutorial12SQL1Click(Sender: TObject);
19003: Procedure tutorial1Click( Sender : TObject)
19004: Procedure tutorial21Click( Sender : TObject)
19005: Procedure tutorial31Click( Sender : TObject)
19006: Procedure tutorial4Click( Sender : TObject)
19007: Procedure Tutorial5Click( Sender : TObject)
19008: procedure Tutorial6Click(Sender: TObject);
19009: procedure Tutorial91Click(Sender: TObject);
19010: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
19011: procedure UniqueString(var str: AnsiString)
19012: procedure UnloadLoadPackage(Module: HMODULE)
```

```
19013: Procedure Unlock
19014: Procedure UNMERGE( MENU : TMAINMENU)
19015: Procedure UnRegisterChanges( Value : TChangeLink)
19016: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
19017: Procedure UnregisterConversionType( const AType : TConvType)
19018: Procedure UnRegisterProvider( Prov : TCustomProvider)
19019: Procedure UPDATE
19020: Procedure UpdateBatch( AffectRecords : TAffectRecords)
19021: Procedure UPDATECURSORPOS
19022: Procedure UpdateFile
19023: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
19024: Procedure UpdateResponse( AResponse : TWebResponse)
19025: Procedure UpdateScrollBar
19026: Procedure UpdateView1Click( Sender : TObject)
19027: procedure Val(const s: string; var n, z: Integer)
19028: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
19029: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
19030: Procedure VariantAdd( const src : Variant; var dst : Variant)
19031: Procedure VariantAnd( const src : Variant; var dst : Variant)
19032: Procedure VariantArrayRedim( var V : Variant; High : Integer)
19033: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
19034: Procedure VariantClear( var V : Variant)
19035: Procedure VariantCpy( const src : Variant; var dst : Variant)
19036: Procedure VariantDiv( const src : Variant; var dst : Variant)
19037: Procedure VariantMod( const src : Variant; var dst : Variant)
19038: Procedure VariantMul( const src : Variant; var dst : Variant)
19039: Procedure VariantOr( const src : Variant; var dst : Variant)
19040: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
19041: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
19042: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
19043: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
19044: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
19045: Procedure VariantShl( const src : Variant; var dst : Variant)
19046: Procedure VariantShr( const src : Variant; var dst : Variant)
19047: Procedure VariantSub( const src : Variant; var dst : Variant)
19048: Procedure VariantXor( const src : Variant; var dst : Variant)
19049: Procedure VarCastError;
19050: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
19051: Procedure VarInvalidOp
19052: Procedure VarInvalidNullOp
19053: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
19054: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
19055: Procedure VarArrayCreateError
19056: Procedure VarResultCheck( AResult : HRESULT);
19057: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
19058: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
19059: Function VarTypeAsText( const AType : TVarType) : string
19060: procedure Voice(const sText: string) //async
19061: procedure Voice2(const sText: string) //sync
19062: Procedure WaitMiliSeconds( AMSec : word)
19063: Procedure WaitMS( AMSec : word)');
19064: procedure WebCamPic(picname: string);  //eg: c:\mypic.png
19065: Procedure WideAppend( var dst : WideString; const src : WideString)
19066: Procedure WideAssign( var dst : WideString; var src : WideString)
19067: Procedure WideDelete( var dst : WideString; index, count : Integer)
19068: Procedure WideFree( var s : WideString)
19069: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
19070: Procedure WideFromPChar( var dst : WideString; src : PChar)
19071: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
19072: Procedure WideSetLength( var dst : WideString; len : Integer)
19073: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
19074: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
19075: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
19076: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19077: Procedure HttpGet(const Url: string; Stream:TStream);
19078: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
19079: Procedure WordWrap1Click( Sender : TObject)
19080: Procedure Write( const AOut : string)
19081: Procedure Write( Socket : TSocket)
19082: procedure Write(S: string);
19083: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
19084: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
19085: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
19086: procedure WriteBuffer(Buffer:String;Count:LongInt)
19087: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
19088: Procedure WriteChar( AValue : Char)
19089: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
19090: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
19091: Procedure WriteFloat( const Section, Name : string; Value : Double)
19092: Procedure WriteHeader( AHeader : TStrings)
19093: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
19094: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
19095: Procedure WriteLn( const AOut : string)
19096: procedure Writeln(s: string);
19097: Procedure WriteLog( const FileName, LogLine : string)
19098: Procedure WriteRFCReply( AReply : TIdRFCReply)
19099: Procedure WriteRFCStrings( AStrings : TStrings)
19100: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
19101: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
```

```
19102: Procedure WriteString( const Section, Ident, Value : String)
19103: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
19104: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
19105: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
19106: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19107: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
19108: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
19109: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
19110: procedure XMLSyntax1Click(Sender: TObject);
19111: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
19112: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
19113: Procedure ZeroFillStream( Stream : TMemoryStream)
19114: procedure XMLSyntax1Click(Sender: TObject);
19115: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
19116: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
19117: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
19118: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
19119: procedure(Sender, Source: TObject;X, Y: Integer)
19120: procedure(Sender, Target: TObject; X, Y: Integer)
19121: procedure(Sender: TObject; ASection, AWidth: Integer)
19122: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
19123: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
19124: procedure(Sender: TObject; var Action: TCloseAction)
19125: procedure(Sender: TObject; var CanClose: Boolean)
19126: procedure(Sender: TObject; var Key: Char);
19127: ProcedureName ProcedureNames ProcedureParametersCursor @
19128:
19129: *************Now Constructors constructor *************
19130:  Size is: 1248 1115 996 628 550 544 501 459 (381)
19131:  Attach( VersionInfoData : Pointer; Size : Integer)
19132:  constructor Create( ABuckets : TBucketListSizes)
19133:  Create( ACallBackWnd : HWND)
19134:  Create( AClient : TCustomTaskDialog)
19135:  Create( AClient : TIdTelnet)
19136:  Create( ACollection : TCollection)
19137:  Create( ACollection : TFavoriteLinkItems)
19138:  Create( ACollection : TTaskDialogButtons)
19139:  Create( AConnection : TIdCustomHTTP)
19140:  Create( ACreateSuspended : Boolean)
19141:  Create( ADataSet : TCustomSQLDataSet)
19142:  CREATE( ADATASET : TDATASET)
19143:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
19144:  Create( AGrid : TCustomDBGrid)
19145:  Create( AGrid : TStringGrid; AIndex : Longint)
19146:  Create( AHTTP : TIdCustomHTTP)
19147:  Create( AListItems : TListItems)
19148:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19149:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
19150:  Create( AOwner : TCommonCalendar)
19151:  Create( AOwner : TComponent)
19152:  CREATE( AOWNER : TCOMPONENT)
19153:  Create( AOwner : TCustomListView)
19154:  Create( AOwner : TCustomOutline)
19155:  Create( AOwner : TCustomRichEdit)
19156:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
19157:  Create( AOwner : TCustomTreeView)
19158:  Create( AOwner : TIdUserManager)
19159:  Create( AOwner : TListItems)
19160:  Create(AOwner:TObj;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
19161:  CREATE( AOWNER : TPERSISTENT)
19162:  Create( AOwner : TPersistent)
19163:  Create( AOwner : TTable)
19164:  Create( AOwner : TTreeNodes)
19165:  Create( AOwner : TWinControl; const ClassName : string)
19166:  Create( AParent : TIdCustomHTTP)
19167:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
19168:  Create( AProvider : TBaseProvider)
19169:  Create( AProvider : TCustomProvider);
19170:  Create( AProvider : TDataSetProvider)
19171:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
19172:  Create( ASocket : TSocket)
19173:  Create( AStrings : TWideStrings)
19174:  Create( AToolBar : TToolBar)
19175:  Create( ATreeNodes : TTreeNodes)
19176:  Create( Autofill : boolean)
19177:  Create( AWebPageInfo : TAbstractWebPageInfo)
19178:  Create( AWebRequest : TWebRequest)
19179:  Create( Collection : TCollection)
19180:  Create( Collection : TIdMessageParts; ABody : TStrings)
19181:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
19182:  Create( Column : TColumn)
19183:  Create( const AConvFamily : TConvFamily; const ADescription : string)
19184:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
19185:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
19186:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
19187:  Create( const ATabSet : TTabSet)
19188:  Create( const Compensate : Boolean)
19189:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
```

```
19190:  Create( const FileName : string)
19191:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
19192:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
19193:  Create( const MaskValue : string)
19194:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
19195:  Create( const Prefix : string)
19196:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
19197:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19198:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
19199:  Create( CoolBar : TCoolBar)
19200:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
19201:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
19202:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
19203:  Create( DBCtrlGrid : TDBCtrlGrid)
19204:  Create( DSTableProducer : TDSTableProducer)
19205:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
19206:  Create( ErrorCode : DBIResult)
19207:  Create( Field : TBlobField; Mode : TBlobStreamMode)
19208:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
19209:  Create( HeaderControl : TCustomHeaderControl)
19210:  Create( HTTPRequest : TWebRequest)
19211:  Create( iStart : integer; sText : string)
19212:  Create( iValue : Integer)
19213:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
19214:  Create( MciErrNo : MCIERROR; const Msg : string)
19215:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
19216:  Create( Message : string; ErrorCode : DBResult)
19217:  Create( Msg : string)
19218:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
19219:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
19220:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
19221:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
19222:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
19223:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
19224:  Create( Owner : TCustomComboBoxEx)
19225:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
19226:  Create( Owner : TPersistent)
19227:  Create( Params : TStrings)
19228:  Create( Size : Cardinal)
19229:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
19230:  Create( StatusBar : TCustomStatusBar)
19231:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
19232:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
19233:  Create(AHandle:Integer)
19234:  Create(AOwner: TComponent); virtual;
19235:  Create(const AURI : string)
19236:  Create(FileName:String;Mode:Word)
19237:  Create(Instance:THandle;ResName:String;ResType:PChar)
19238:  Create(Stream : TStream)
19239:  Create1( ADataset : TDataset);
19240:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
19241:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
19242:  Create2( Other : TObject);
19243:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
19244:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
19245:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
19246:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
19247:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
19248:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
19249:  CreateRes( Ident : Integer);
19250:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
19251:  CreateRes( ResStringRec : PResStringRec);
19252:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
19253:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
19254:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
19255:  CreateSize( AWidth, AHeight : Integer)
19256:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
19257:
19258: ------------------------------------------------------------------------------
19259: unit uPSI_MathMax;
19260: ------------------------------------------------------------------------------
19261:  CONSTS
19262:   Bernstein: Float = 0.28016949902386913303643649123 07;  // Bernstein constant
19263:   Crt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
19264:   Crt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
19265:   Crt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
19266:   Crt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
19267:   CrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
19268:   Catalan: Float  = 0.91596559417721901505460351493 24;  // Catalan constant
19269:   PiJ:   Float      = 3.14159265358979323846264338327 95;  // PI
19270:   PI:  Extended =  3.14159265358979323846264338327 95);
19271:   PiOn2: Float     = 1.57079632679489661923132169163 98;  // PI / 2
19272:   PiOn3: Float     = 1.04719755119659774615421446109 32;  // PI / 3
19273:   PiOn4: Float     = 0.78539816339744830961566084581 988; // PI / 4
19274:   Sqrt2: Float     = 1.41421356237309504880168872420 97;  // Sqrt(2)
```

```
19275:   Sqrt3: Float     = 1.732050807568877293527446341505 9;  // Sqrt(3)
19276:   Sqrt5: Float     = 2.236067977499789696409173668731 3;  // Sqrt(5)
19277:   Sqrt10: Float    = 3.162277660168379331998893544432 7;  // Sqrt(10)
19278:   SqrtPi: Float    = 1.772453850905516027298167483341 1;  // Sqrt(PI)
19279:   Sqrt2Pi: Float   = 2.506628274631000502415765284811;   // Sqrt(2 * PI)
19280:   TwoPi: Float     = 6.283185307179586476925286766559;   // 2 * PI
19281:   ThreePi: Float   = 9.424777960769379715387930149838 5;  // 3 * PI
19282:   Ln2: Float       = 0.693147180559945309417232121458 18; // Ln(2)
19283:   Ln10: Float      = 2.302585092994045684017991454684 4;  // Ln(10)
19284:   LnPi: Float      = 1.144729885849400174143427351353 1;  // Ln(PI)
19285:   Log2J: Float     = 0.301029995663981195213738894724 49; // Log10(2)
19286:   Log3: Float      = 0.477121254719662437295027903255 12; // Log10(3)
19287:   LogPi: Float     = 0.497149872694133854351268288290 9;  // Log10(PI)
19288:   LogE: Float      = 0.434294481903251827651128918916 61; // Log10(E)
19289:   E: Float         = 2.718281828459045235360287471352 7;  // Natural constant
19290:   hLn2Pi: Float    = 0.918938533204672741780329736405 62; // Ln(2*PI)/2
19291:   inv2Pi: Float    = 0.159154943091895335768883763372514362034459645740 46; // 0.5/Pi
19292:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
19293:   GoldenMean: Float   = 1.618033988749894848204586834365638;  // GoldenMean
19294:   EulerMascheroni: Float = 0.577215664901532860606512090082 4;  // Euler GAMMA
19295:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
19296:   StDelta         : Extended = 0.00001;  {delta for difference equations}
19297:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
19298:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
19299:
19300: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
19301: begin
19302:   MetersPerInch = 0.0254; // [1]
19303:   MetersPerFoot = MetersPerInch * 12;
19304:   MetersPerYard = MetersPerFoot * 3;
19305:   MetersPerMile = MetersPerFoot * 5280;
19306:   MetersPerNauticalMiles = 1852;
19307:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
19308:   MetersPerLightSecond = 2.99792458E8; // [5]
19309:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
19310:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
19311:   MetersPerCubit = 0.4572; // [6][7]
19312:   MetersPerFathom = MetersPerFoot * 6;
19313:   MetersPerFurlong = MetersPerYard * 220;
19314:   MetersPerHand = MetersPerInch * 4;
19315:   MetersPerPace = MetersPerInch * 30;
19316:   MetersPerRod = MetersPerFoot * 16.5;
19317:   MetersPerChain = MetersPerRod * 4;
19318:   MetersPerLink = MetersPerChain / 100;
19319:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
19320:   MetersPerPica = MetersPerPoint * 12;
19321:
19322:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
19323:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
19324:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
19325:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
19326:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
19327:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
19328:
19329:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
19330:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
19331:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
19332:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
19333:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
19334:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
19335:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
19336:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
19337:
19338:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
19339:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
19340:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
19341:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
19342:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
19343:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
19344:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
19345:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
19346:
19347:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
19348:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
19349:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
19350:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
19351:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
19352:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
19353:
19354:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
19355:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
19356:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
19357:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
19358:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
19359:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
19360:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
19361:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
19362:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
19363:
```

```
19364:   GramsPerPound = 453.59237; // [1][7]
19365:   GramsPerDrams = GramsPerPound / 256;
19366:   GramsPerGrains = GramsPerPound / 7000;
19367:   GramsPerTons = GramsPerPound * 2000;
19368:   GramsPerLongTons = GramsPerPound * 2240;
19369:   GramsPerOunces = GramsPerPound / 16;
19370:   GramsPerStones = GramsPerPound * 14;
19371:
19372:   MaxAngle 9223372036854775808.0;
19373:   MaxTanH 5678.26170314707197474596553898 54);
19374:   MaxFactorial( 1754);
19375:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
19376:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
19377:   MaxTanH( 354.89135644669199842162284618659);
19378:   MaxFactorial'LongInt'( 170);
19379:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
19380:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
19381:   MaxTanH( 44.361419555836499802702855773323);
19382:   MaxFactorial'LongInt'( 33);
19383:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
19384:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
19385:   PiExt( 3.14159265358979323846264338327 95);
19386:   RatioDegToRad( PiExt / 180.0);
19387:   RatioGradToRad( PiExt / 200.0);
19388:   RatioDegToGrad( 200.0 / 180.0);
19389:   RatioGradToDeg( 180.0 / 200.0);
19390:   Crc16PolynomCCITT'LongWord $1021);
19391:  Crc16PolynomIBM'LongWord $8005);
19392:  Crc16Bits'LongInt'( 16);
19393:  Crc16Bytes'LongInt'( 2);
19394:  Crc16HighBit'LongWord $8000);
19395:  NotCrc16HighBit','LongWord $7FFF);
19396:   Crc32PolynomIEEE','LongWord $04C11DB7);
19397:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
19398:  Crc32Koopman','LongWord $741B8CD7);
19399:  Crc32Bits','LongInt'( 32);
19400:  Crc32Bytes','LongInt'( 4);
19401:  Crc32HighBit','LongWord $80000000);
19402:  NotCrc32HighBit','LongWord $7FFFFFFF);
19403:
19404:   MinByte       = Low(Byte);
19405:   MaxByte       = High(Byte);
19406:   MinWord       = Low(Word);
19407:   MaxWord       = High(Word);
19408:   MinShortInt   = Low(ShortInt);
19409:   MaxShortInt   = High(ShortInt);
19410:   MinSmallInt   = Low(SmallInt);
19411:   MaxSmallInt   = High(SmallInt);
19412:   MinLongWord   = LongWord(Low(LongWord));
19413:   MaxLongWord   = LongWord(High(LongWord));
19414:   MinLongInt    = LongInt(Low(LongInt));
19415:   MaxLongInt    = LongInt(High(LongInt));
19416:   MinInt64      = Int64(Low(Int64));
19417:   MaxInt64      = Int64(High(Int64));
19418:   MinInteger    = Integer(Low(Integer));
19419:   MaxInteger    = Integer(High(Integer));
19420:   MinCardinal   = Cardinal(Low(Cardinal));
19421:   MaxCardinal   = Cardinal(High(Cardinal));
19422:   MinNativeUInt = NativeUInt(Low(NativeUInt));
19423:   MaxNativeUInt = NativeUInt(High(NativeUInt));
19424:   MinNativeInt  = NativeInt(Low(NativeInt));
19425:   MaxNativeInt  = NativeInt(High(NativeInt));
19426:  Function CosH( const Z : Float) : Float;
19427:  Function SinH( const Z : Float) : Float;
19428:  Function TanH( const Z : Float) : Float;
19429:
19430:
19431:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
19432:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
19433:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
19434:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
19435:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
19436:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
19437:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
19438:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
19439:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
19440:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
19441:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
19442:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
19443:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
19444:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
19445:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
19446:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
19447:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
19448:   MaxLog = 709.7827128933840;
19449:   MinLog = -708.3964185322641;
19450:   MaxFac = 170;
19451:   MaxGam = 171.624376956302;
19452:   MaxLgm = 2.556348E+305;
```

```
19453:   SingleCompareDelta   = 1.0E-34;
19454:   DoubleCompareDelta   = 1.0E-280;
19455:   {$IFDEF CLR}
19456:   ExtendedCompareDelta = DoubleCompareDelta;
19457:   {$ELSE}
19458:   ExtendedCompareDelta = 1.0E-4400;
19459:   {$ENDIF}
19460:   Bytes1KB  = 1024;
19461:   Bytes1MB  = 1024 * Bytes1KB;
19462:   Bytes1GB  = 1024 * Bytes1MB;
19463:   Bytes64KB = 64 * Bytes1KB;
19464:   Bytes64MB = 64 * Bytes1MB;
19465:   Bytes2GB  = 2 * LongWord(Bytes1GB);
19466:     clBlack32', $FF000000 ));
19467:     clDimGray32', $FF3F3F3F ));
19468:     clGray32', $FF7F7F7F ));
19469:     clLightGray32', $FFBFBFBF ));
19470:     clWhite32', $FFFFFFFF ));
19471:     clMaroon32', $FF7F0000 ));
19472:     clGreen32', $FF007F00 ));
19473:     clOlive32', $FF7F7F00 ));
19474:     clNavy32', $FF00007F ));
19475:     clPurple32', $FF7F007F ));
19476:     clTeal32', $FF007F7F ));
19477:     clRed32', $FFFF0000 ));
19478:     clLime32', $FF00FF00 ));
19479:     clYellow32', $FFFFFF00 ));
19480:     clBlue32', $FF0000FF ));
19481:     clFuchsia32', $FFFF00FF ));
19482:     clAqua32', $FF00FFFF ));
19483:     clAliceBlue32', $FFF0F8FF ));
19484:     clAntiqueWhite32', $FFFAEBD7 ));
19485:     clAquamarine32', $FF7FFFD4 ));
19486:     clAzure32', $FFF0FFFF ));
19487:     clBeige32', $FFF5F5DC ));
19488:     clBisque32', $FFFFE4C4 ));
19489:     clBlancheDalmond32', $FFFFEBCD ));
19490:     clBlueViolet32', $FF8A2BE2 ));
19491:     clBrown32', $FFA52A2A ));
19492:     clBurlyWood32', $FFDEB887 ));
19493:     clCadetblue32', $FF5F9EA0 ));
19494:     clChartReuse32', $FF7FFF00 ));
19495:     clChocolate32', $FFD2691E ));
19496:     clCoral32', $FFFF7F50 ));
19497:     clCornFlowerBlue32', $FF6495ED ));
19498:     clCornSilk32', $FFFFF8DC ));
19499:     clCrimson32', $FFDC143C ));
19500:     clDarkBlue32', $FF00008B ));
19501:     clDarkCyan32', $FF008B8B ));
19502:     clDarkGoldenRod32', $FFB8860B ));
19503:     clDarkGray32', $FFA9A9A9 ));
19504:     clDarkGreen32', $FF006400 ));
19505:     clDarkGrey32', $FFA9A9A9 ));
19506:     clDarkKhaki32', $FFBDB76B ));
19507:     clDarkMagenta32', $FF8B008B ));
19508:     clDarkOliveGreen32', $FF556B2F ));
19509:     clDarkOrange32', $FFFF8C00 ));
19510:     clDarkOrchid32', $FF9932CC ));
19511:     clDarkRed32', $FF8B0000 ));
19512:     clDarkSalmon32', $FFE9967A ));
19513:     clDarkSeaGreen32', $FF8FBC8F ));
19514:     clDarkSlateBlue32', $FF483D8B ));
19515:     clDarkSlateGray32', $FF2F4F4F ));
19516:     clDarkSlateGrey32', $FF2F4F4F ));
19517:     clDarkTurquoise32', $FF00CED1 ));
19518:     clDarkViolet32', $FF9400D3 ));
19519:     clDeepPink32', $FFFF1493 ));
19520:     clDeepSkyBlue32', $FF00BFFF ));
19521:     clDodgerBlue32', $FF1E90FF ));
19522:     clFireBrick32', $FFB22222 ));
19523:     clFloralWhite32', $FFFFFAF0 ));
19524:     clGainsBoro32', $FFDCDCDC ));
19525:     clGhostWhite32', $FFF8F8FF ));
19526:     clGold32', $FFFFD700 ));
19527:     clGoldenRod32', $FFDAA520 ));
19528:     clGreenYellow32', $FFADFF2F ));
19529:     clGrey32', $FF808080 ));
19530:     clHoneyDew32', $FFF0FFF0 ));
19531:     clHotPink32', $FFFF69B4 ));
19532:     clIndianRed32', $FFCD5C5C ));
19533:     clIndigo32', $FF4B0082 ));
19534:     clIvory32', $FFFFFFF0 ));
19535:     clKhaki32', $FFF0E68C ));
19536:     clLavender32', $FFE6E6FA ));
19537:     clLavenderBlush32', $FFFFF0F5 ));
19538:     clLawnGreen32', $FF7CFC00 ));
19539:     clLemonChiffon32', $FFFFFACD ));
19540:     clLightBlue32', $FFADD8E6 ));
19541:     clLightCoral32', $FFF08080 ));
```

```
19542:     clLightCyan32', $FFE0FFFF ));
19543:     clLightGoldenRodYellow32', $FFFAFAD2 ));
19544:     clLightGreen32', $FF90EE90 ));
19545:     clLightGrey32', $FFD3D3D3 ));
19546:     clLightPink32', $FFFFB6C1 ));
19547:     clLightSalmon32', $FFFFA07A ));
19548:     clLightSeagreen32', $FF20B2AA ));
19549:     clLightSkyblue32', $FF87CEFA ));
19550:     clLightSlategray32', $FF778899 ));
19551:     clLightSlategrey32', $FF778899 ));
19552:     clLightSteelblue32', $FFB0C4DE ));
19553:     clLightYellow32', $FFFFFFE0 ));
19554:     clLtGray32', $FFC0C0C0 ));
19555:     clMedGray32', $FFA0A0A4 ));
19556:     clDkGray32', $FF808080 ));
19557:     clMoneyGreen32', $FFC0DCC0 ));
19558:     clLegacySkyBlue32', $FFA6CAF0 ));
19559:     clCream32', $FFFFFBF0 ));
19560:     clLimeGreen32', $FF32CD32 ));
19561:     clLinen32', $FFFAF0E6 ));
19562:     clMediumAquamarine32', $FF66CDAA ));
19563:     clMediumBlue32', $FF0000CD ));
19564:     clMediumOrchid32', $FFBA55D3 ));
19565:     clMediumPurple32', $FF9370DB ));
19566:     clMediumSeaGreen32', $FF3CB371 ));
19567:     clMediumSlateBlue32', $FF7B68EE ));
19568:     clMediumSpringGreen32', $FF00FA9A ));
19569:     clMediumTurquoise32', $FF48D1CC ));
19570:     clMediumVioletRed32', $FFC71585 ));
19571:     clMidnightBlue32', $FF191970 ));
19572:     clMintCream32', $FFF5FFFA ));
19573:     clMistyRose32', $FFFFE4E1 ));
19574:     clMoccasin32', $FFFFE4B5 ));
19575:     clNavajoWhite32', $FFFFDEAD ));
19576:     clOldLace32', $FFFDF5E6 ));
19577:     clOliveDrab32', $FF6B8E23 ));
19578:     clOrange32', $FFFFA500 ));
19579:     clOrangeRed32', $FFFF4500 ));
19580:     clOrchid32', $FFDA70D6 ));
19581:     clPaleGoldenRod32', $FFEEE8AA ));
19582:     clPaleGreen32', $FF98FB98 ));
19583:     clPaleTurquoise32', $FFAFEEEE ));
19584:     clPaleVioletred32', $FFDB7093 ));
19585:     clPapayaWhip32', $FFFFEFD5 ));
19586:     clPeachPuff32', $FFFFDAB9 ));
19587:     clPeru32', $FFCD853F ));
19588:     clPlum32', $FFDDA0DD ));
19589:     clPowderBlue32', $FFB0E0E6 ));
19590:     clRosyBrown32', $FFBC8F8F ));
19591:     clRoyalBlue32', $FF4169E1 ));
19592:     clSaddleBrown32', $FF8B4513 ));
19593:     clSalmon32', $FFFA8072 ));
19594:     clSandyBrown32', $FFF4A460 ));
19595:     clSeaGreen32', $FF2E8B57 ));
19596:     clSeaShell32', $FFFFF5EE ));
19597:     clSienna32', $FFA0522D ));
19598:     clSilver32', $FFC0C0C0 ));
19599:     clSkyblue32', $FF87CEEB ));
19600:     clSlateBlue32', $FF6A5ACD ));
19601:     clSlateGray32', $FF708090 ));
19602:     clSlateGrey32', $FF708090 ));
19603:     clSnow32', $FFFFFAFA ));
19604:     clSpringgreen32', $FF00FF7F ));
19605:     clSteelblue32', $FF4682B4 ));
19606:     clTan32', $FFD2B48C ));
19607:     clThistle32', $FFD8BFD8 ));
19608:     clTomato32', $FFFF6347 ));
19609:     clTurquoise32', $FF40E0D0 ));
19610:     clViolet32', $FFEE82EE ));
19611:     clWheat32', $FFF5DEB3 ));
19612:     clWhitesmoke32', $FFF5F5F5 ));
19613:     clYellowgreen32', $FF9ACD32 ));
19614:     clTrWhite32', $7FFFFFFF ));
19615:     clTrBlack32', $7F000000 ));
19616:     clTrRed32', $7FFF0000 ));
19617:     clTrGreen32', $7F00FF00 ));
19618:     clTrBlue32', $7F0000FF ));
19619:   // Fixed point math constants
19620:   FixedOne = $10000;  FixedHalf = $7FFF;
19621:   FixedPI  = Round(PI * FixedOne);
19622:   FixedToFloat = 1/FixedOne;
19623:
19624:  Special Types
19625: ************************************************
19626:   type Complex = record
19627:     X, Y : Float;
19628:   end;
19629:   type TVector     = array of Float;
19630:   TIntVector  = array of Integer;
```

```
19631:   TCompVector = array of Complex;
19632:   TBoolVector = array of Boolean;
19633:   TStrVector  = array of String;
19634:   TMatrix     = array of TVector;
19635:   TIntMatrix  = array of TIntVector;
19636:   TCompMatrix = array of TCompVector;
19637:   TBoolMatrix = array of TBoolVector;
19638:   TStrMatrix  = array of TStrVector;
19639:   TByteArray = array[0..32767] of byte; !
19640:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
19641:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
19642:   T2StringArray = array of array of string;
19643:   T2IntegerArray = array of array of integer;
19644:   AddTypeS('INT_PTR', 'Integer
19645:   AddTypeS('LONG_PTR', 'Integer
19646:   AddTypeS('UINT_PTR', 'Cardinal
19647:   AddTypeS('ULONG_PTR', 'Cardinal
19648:   AddTypeS('DWORD_PTR', 'ULONG_PTR
19649:   TIntegerDynArray', 'array of Integer
19650:   TCardinalDynArray', 'array of Cardinal
19651:   TWordDynArray', 'array of Word
19652:   TSmallIntDynArray', 'array of SmallInt
19653:   TByteDynArray', 'array of Byte
19654:   TShortIntDynArray', 'array of ShortInt
19655:   TInt64DynArray', 'array of Int64
19656:   TLongWordDynArray', 'array of LongWord
19657:   TSingleDynArray', 'array of Single
19658:   TDoubleDynArray', 'array of Double
19659:   TBooleanDynArray', 'array of Boolean
19660:   TStringDynArray', 'array of string
19661:   TWideStringDynArray', 'array of WideString
19662:   TDynByteArray     = array of Byte;
19663:   TDynShortintArray = array of Shortint;
19664:   TDynSmallintArray = array of Smallint;
19665:   TDynWordArray     = array of Word;
19666:   TDynIntegerArray  = array of Integer;
19667:   TDynLongintArray  = array of Longint;
19668:   TDynCardinalArray = array of Cardinal;
19669:   TDynInt64Array    = array of Int64;
19670:   TDynExtendedArray = array of Extended;
19671:   TDynDoubleArray   = array of Double;
19672:   TDynSingleArray   = array of Single;
19673:   TDynFloatArray    = array of Float;
19674:   TDynPointerArray  = array of Pointer;
19675:   TDynStringArray   = array of string;
19676:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
19677:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
19678:   TSynSearchOptions = set of TSynSearchOption;
19679:
19680:
19681: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
19682: ------------------------------------------------------------------------------
19683: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
19684: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
19685: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
19686: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19687: function CheckStringSum(vstring: string): integer;
19688: function HexToInt(HexNum: string): LongInt;
19689: function IntToBin(Int: Integer): String;
19690: function BinToInt(Binary: String): Integer;
19691: function HexToBin(HexNum: string): string; external2
19692: function BinToHex(Binary: String): string;
19693: function IntToFloat(i: Integer): double;
19694: function AddThousandSeparator(S: string; myChr: Char): string;
19695: function Max3(const X,Y,Z: Integer): Integer;
19696: procedure Swap(var X,Y: char); // faster without inline
19697: procedure ReverseString(var S: String);
19698: function CharToHexStr(Value: Char): string;
19699: function CharToUniCode(Value: Char): string;
19700: function Hex2Dec(Value: Str002): Byte;
19701: function HexStrCodeToStr(Value: string): string;
19702: function HexToStr(i: integer; value: string): string;
19703: function UniCodeToStr(Value: string): string;
19704: function CRC16(statement: string): string;
19705: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
19706: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
19707: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19708: Procedure ExecuteCommand(executeFile, paramstring: string);
19709: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
19710: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
19711: procedure SearchAndOpenDoc(vfilenamepath: string);
19712: procedure ShowInterfaces(myFile: string);
19713: function Fact2(av: integer): extended;
19714: Function BoolToStr(B: Boolean): string;
19715: Function GCD(x, y : LongInt) : LongInt;
19716: function LCM(m,n: longint): longint;
19717: function GetASCII: string;
19718: function GetItemHeight(Font: TFont): Integer;
19719: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
```

```
19720: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
19721: function getHINSTANCE: longword;
19722: function getHMODULE: longword;
19723: function GetASCII: string;
19724: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
19725: function WordIsOk(const AWord: string; var VW: Word): boolean;
19726: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
19727: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
19728: function SafeStr(const s: string): string;
19729: function ExtractUrlPath(const FileName: string): string;
19730: function ExtractUrlName(const FileName: string): string;
19731: function IsInternet: boolean;
19732: function RotateLeft1Bit_u32( Value: uint32): uint32;
19733: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats: Bool);
19734: procedure getEnvironmentInfo;
19735: procedure AntiFreeze;
19736: function GetCPUSpeed: Double;
19737: function IsVirtualPcGuest : Boolean;
19738: function IsVmWareGuest : Boolean;
19739: procedure StartSerialDialog;
19740: function IsWoW64: boolean;
19741: function IsWow64String(var s: string): Boolean;
19742: procedure StartThreadDemo;
19743: Function RGB(R,G,B: Byte): TColor;
19744: Function Sendln(amess: string): boolean;
19745: Procedure maXbox;
19746: Function AspectRatio(aWidth, aHeight: Integer): String;
19747: function wget(aURL, afile: string): boolean;
19748: procedure PrintList(Value: TStringList);
19749: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
19750: procedure getEnvironmentInfo;
19751: procedure AntiFreeze;
19752: function getBitmap(apath: string): TBitmap;
19753: procedure ShowMessageBig(const aText : string);
19754: function YesNoDialog(const ACaption, AMsg: string): boolean;
19755: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
19756: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19757: //function myStrToBytes(const Value: String): TBytes;
19758: //function myBytesToStr(const Value: TBytes): String;
19759: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19760: function getBitmap(apath: string): TBitmap;
19761: procedure ShowMessageBig(const aText : string);
19762: Function StrToBytes(const Value: String): TBytes;
19763: Function BytesToStr(const Value: TBytes): String;
19764: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19765: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
19766: function FindInPaths(const fileName, paths : String) : String;
19767: procedure initHexArray(var hexn: THexArray);
19768: function josephusG(n,k: integer; var graphout: string): integer;
19769: function isPowerof2(num: int64): boolean;
19770: function powerOf2(exponent: integer): int64;
19771: function getBigPI: string;
19772: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
19773:  function GetASCIILine: string;
19774: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
19775:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
19776: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19777: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19778: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
19779: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
19780: function isKeypressed: boolean;
19781: function Keypress: boolean;
19782: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19783: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
19784: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
19785: function GetOSName: string;
19786: function GetOSVersion: string;
19787: function GetOSNumber: string;
19788: function getEnvironmentString: string;
19789: procedure StrReplace(var Str: String; Old, New: String);
19790: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
19791: function getTeamViewerID: string;
19792: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
19793: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
19794: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19795: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
19796: function StartSocketService: Boolean;
19797: procedure StartSocketServiceForm;
19798: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
19799: function GetFileList1(apath: string): TStringlist;
19800: procedure LetFileList(FileList: TStringlist; apath: string);
19801: procedure StartWeb(aurl: string);
19802: function GetTodayFiles(startdir, amask: string): TStringlist;
19803: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
19804: function JavahashCode(val: string): Integer;
19805: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19806: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
```

```
19807: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
19808: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
19809: Procedure ConvertToGray(Cnv: TCanvas);
19810: function GetFileDate(aFile:string; aWithTime:Boolean):string;
19811: procedure ShowMemory;
19812: function ShowMemory2: string;
19813: function getHostIP: string;
19814: procedure ShowBitmap(bmap: TBitmap);
19815: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
19816: function CreateDBGridForm(dblist: TStringList): TListbox;
19817: function isService: boolean;
19818: function isApplication: boolean;
19819: function isTerminalSession: boolean;
19820:
19821:
19822: // News of 3.9.8 up
19823: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19824: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19825: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19826: JvChart - TJvChart Component - 2009 Public
19827: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
19828: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
19829: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
19830: DMath DLL included incl. Demos
19831: Interface Navigator menu/View/Intf Navigator
19832: Unit Explorer menu/Debug/Units Explorer
19833: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
19834: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
19835: Script History to 9 Files WebServer light /Options/Addons/WebServer
19836: Full Text Finder, JVSimLogic Simulator Package
19837: Halt-Stop Program in Menu, WebServer2, Stop Event ,
19838: Conversion Routines, Prebuild Forms, CodeSearch
19839: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19840: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19841: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19842: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
19843: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
19844: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
19845: IDE Reflection API, Session Service Shell S3
19846: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
19847: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
19848: arduino map() function, PMRandom Generator
19849: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
19850: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
19851: REST Test Lib, Multilang Component, Forth Interpreter
19852: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
19853: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
19854: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19855: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19856: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
19857: QRCode Service, add more CFunctions like CDateTime of Synapse
19858: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
19859: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
19860: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
19861: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
19862: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
19863: BOLD Package, Indy Package5, maTRIx. MATHEMAX
19864: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
19865: emax layers: system-package-component-unit-class-function-block
19866: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
19867: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
19868: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
19869: OpenGL Game Demo: ..Options/Add Ons/Reversi
19870: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
19871: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
19872: 7% performance gain (hot spot profiling)
19873: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
19874: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
19875: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
19876: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools, Zeus
19877:
19878: add routines in 3.9.7.5
19879: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
19880: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
19881: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
19882: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
19883: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
19884: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
19885: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
19886:
19887: ////////////////////////// TestUnits ////////////////////////////
19888:   SelftestPEM;
19889:   SelfTestCFundamentUtils;
19890:   SelfTestCFileUtils;
19891:   SelfTestCDateTime;
19892:   SelfTestCTimer;
19893:   SelfTestCRandom;
19894:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
19895:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
```

```
19896:
19897:   // Note: There's no need for installing a client certificate in the
19898:   //       webbrowser. The server asks the webbrowser to send a certificate but
19899:   //       if nothing is installed the software will work because the server
19900:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
19901:   //       file: c_cacert.p12  password: c_cakey
19902:
19903:   TGraphicControl = class(TControl)
19904:   private
19905:     FCanvas: TCanvas;
19906:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19907:   protected
19908:     procedure Paint; virtual;
19909:     property Canvas: TCanvas read FCanvas;
19910:   public
19911:     constructor Create(AOwner: TComponent); override;
19912:     destructor Destroy; override;
19913:   end;
19914:
19915:   TCustomControl = class(TWinControl)
19916:   private
19917:     FCanvas: TCanvas;
19918:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19919:   protected
19920:     procedure Paint; virtual;
19921:     procedure PaintWindow(DC: HDC); override;
19922:     property Canvas: TCanvas read FCanvas;
19923:   public
19924:     constructor Create(AOwner: TComponent); override;
19925:     destructor Destroy; override;
19926:   end;
19927:     RegisterPublishedProperties;
19928:     ('ONCHANGE', 'TNotifyEvent', iptrw);
19929:     ('ONCLICK', 'TNotifyEvent', iptrw);
19930:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
19931:     ('ONENTER', 'TNotifyEvent', iptrw);
19932:     ('ONEXIT', 'TNotifyEvent', iptrw);
19933:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
19934:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
19935:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
19936:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
19937:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
19938: //*************************************************************************
19939:   // To stop the while loop, click on Options/Show Include (boolean switch)!
19940:   Control a loop in a script with a form event:
19941:   IncludeON;   //control the while loop
19942:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
19943:   repeat {for it:= 1 to n do} until is keypressed  //keypress in output window below (memo2)
19944:
19945: //-------------------------------------------------------------------------
19946: //*************mX4 ini-file Configuration*********************************
19947: //-------------------------------------------------------------------------
19948:   using config file maxboxdef.ini       menu/Help/Config File
19949:
19950: //*** Definitions for maXbox mX3 ***
19951: [FORM]
19952: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
19953: FONTSIZE=14
19954: EXTENSION=txt
19955: SCREENX=1386
19956: SCREENY=1077
19957: MEMHEIGHT=350
19958: PRINTFONT=Courier New //GUI Settings
19959: LINENUMBERS=Y   //line numbers at gutter in editor at left side
19960: EXCEPTIONLOG=Y  //store excepts + success in 2 log files see below! –menu Debug/Show Last Exceptions
19961: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
19962: BOOTSCRIPT=Y    //enabling load a boot script
19963: MEMORYREPORT=Y  //shows memory report on closing maXbox
19964: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
19965: NAVIGATOR=N     //shows function list at the right side of editor
19966: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
19967: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
19968: [WEB]
19969: IPPORT=8080     //for internal webserver – menu /Options/Add Ons/WebServer
19970: IPHOST=192.168.1.53
19971: ROOTCERT=filepathY
19972: SCERT=filepathY
19973: RSAKEY=filepathY
19974: VERSIONCHECK=Y
19975:
19976: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
19977: Also possible to set report memory in script to override ini setting
19978: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19979:
19980: After Change the ini file you can reload the file with ../Help/Config Update
19981:
19982: //-------------------------------------------------------------------------
19983: //*************mX4 maildef.ini ini-file Configuration*********************
19984: //-------------------------------------------------------------------------
```

```
19985: //*** Definitions for maXMail ***
19986: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
19987: [MAXMAIL]
19988: HOST=getmail.softwareschule.ch
19989: USER=mailusername
19990: PASS=password
19991: PORT=110
19992: SSL=Y
19993: BODY=Y
19994: LAST=5
19995:
19996: ADO Connection String:
19997: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
19998: \452_dbtreeview2access.txt
19999: program TestDbTreeViewMainForm2_ACCESS;
20000:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
20001:                    +Exepath+'\examples\detail.mdb;Persist Security Info=False';
20002:
20003: OpenSSL Lib:  unit ssl_openssl_lib;
20004: {$IFDEF CIL}
20005: const
20006:   {$IFDEF LINUX}
20007:   DLLSSLName = 'libssl.so';
20008:   DLLUtilName = 'libcrypto.so';
20009:   {$ELSE}
20010:   DLLSSLName = 'ssleay32.dll';
20011:   DLLUtilName = 'libeay32.dll';
20012:   {$ENDIF}
20013: {$ELSE}
20014: var
20015:   {$IFNDEF MSWINDOWS}
20016:     {$IFDEF DARWIN}
20017:     DLLSSLName: string = 'libssl.dylib';
20018:     DLLUtilName: string = 'libcrypto.dylib';
20019:     {$ELSE}
20020:     DLLSSLName: string = 'libssl.so';
20021:     DLLUtilName: string = 'libcrypto.so';
20022:     {$ENDIF}
20023:   {$ELSE}
20024:   DLLSSLName: string = 'ssleay32.dll';
20025:   DLLSSLName2: string = 'libssl32.dll';
20026:   DLLUtilName: string = 'libeay32.dll';
20027:   {$ENDIF}
20028: {$ENDIF}
20029:
20030:
20031: //-------------------------------------------------------------------------
20032: //**************mX4 Macro Tags  ********************************************
20033: //-------------------------------------------------------- ----------------
20034:
20035:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
     E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
20036:
20037: //Tag Macros
20038:
20039:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
20040:
20041: //Tag Macros
20042: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
20043: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
20044: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
20045: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
20046: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
20047: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
20048: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
20049: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
20050: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
20051:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
20052: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
20053: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20054:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
20055: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
20056:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
20057: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
20058:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
20059:
20060: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
20061:
20062: //Replace Macros
20063:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
20064:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
20065:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
20066:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
20067:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
20068:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
20069:
20070:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
20071:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
20072: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
```

```
20073:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
20074:
20075: //------------------------------------------------------------------------------
20076: //**************mX4 ToDo List Tags  ../Help/ToDo List************************
20077: //------------------------------------------------------------ ---------------------
20078:
20079:     while I < sl.Count do begin
20080: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
20081:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
20082:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
20083:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
20084:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
20085:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
20086:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
20087:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
20088:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
20089:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
20090:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
20091:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
20092:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
20093:       Inc(I);
20094:     end;
20095:
20096: //------------------------------------------------------------------------------
20097: //**************mX4 Public  Tools API ***************************************
20098: //------------------------------------------------------------------------------
20099:   file : unit uPSI_fMain.pas;               {$OTAP} Open Tools API Catalog
20100:  // Those functions concern the editor and preprocessor, all of the IDE
20101:  Example: Call it with maxform1.Info1Click(self)
20102:  Note: Call all Methods with maxForm1., e.g.:
20103:                        maxForm1.ShellStyle1Click(self);
20104:
20105: procedure SIRegister_fMain(CL: TPSPascalCompiler);
20106: begin
20107:  Const('BYTECODE','String 'bytecode.txt'
20108:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
20109:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
20110:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
20111:  Const('PSINC','String PS Includes (*.inc)|*.INC
20112:  Const('DEFFILENAME','String 'firstdemo.txt
20113:  Const('DEFINIFILE','String 'maxboxdef.ini
20114:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
20115:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
20116:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
20117:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
20118:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
20119:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
20120:  Const('INCLUDEBOX','String 'pas_includebox.inc
20121:  Const('BOOTSCRIPT','String 'maxbootscript.txt
20122:  Const('MBVERSION','String '3.9.9.98
20123:  Const('VERSION','String'3.9.9.98
20124:  Const('MBVER','String '399
20125:  Const('MBVERI','Integer'(399);
20126:  Const('MBVERIALL','Integer'(39998);
20127:  Const('EXENAME','String 'maXbox3.exe
20128:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
20129:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
20130:  Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
20131:  Const('MXINTERNETCHECK','String 'www.ask.com
20132:  Const('MXMAIL','String 'max@kleiner.com
20133:  Const('TAB','Char #$09);
20134:  Const('CODECOMPLETION','String 'bds_delphi.dci
20135:  SIRegister_TMaxForm1(CL);
20136: end;
20137:
20138:   with FindClass('TForm'),'TMaxForm1') do begin
20139:     memo2', 'TMemo', iptrw);
20140:     memo1', 'TSynMemo', iptrw);
20141:     CB1SCList', 'TComboBox', iptrw);
20142:     mxNavigator', 'TComboBox', iptrw);
20143:     IPHost', 'string', iptrw);
20144:     IPPort', 'integer', iptrw);
20145:     COMPort', 'integer', iptrw);      //3.9.6.4
20146:     Splitter1', 'TSplitter', iptrw);
20147:     PSScript', 'TPSScript', iptrw);
20148:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
20149:     MainMenu1', 'TMainMenu', iptrw);
20150:     Program1', 'TMenuItem', iptrw);
20151:     Compile1', 'TMenuItem', iptrw);
20152:     Files1', 'TMenuItem', iptrw);
20153:     open1', 'TMenuItem', iptrw);
20154:     Save2', 'TMenuItem', iptrw);
20155:     Options1', 'TMenuItem', iptrw);
20156:     Savebefore1', 'TMenuItem', iptrw);
20157:     Largefont1', 'TMenuItem', iptrw);
20158:     sBytecode1', 'TMenuItem', iptrw);
20159:     Saveas3', 'TMenuItem', iptrw);
20160:     Clear1', 'TMenuItem', iptrw);
20161:     Slinenumbers1', 'TMenuItem', iptrw);
```

```
20162:     About1', 'TMenuItem', iptrw);
20163:     Search1', 'TMenuItem', iptrw);
20164:     SynPasSyn1', 'TSynPasSyn', iptrw);
20165:     memo1', 'TSynMemo', iptrw);
20166:     SynEditSearch1', 'TSynEditSearch', iptrw);
20167:     WordWrap1', 'TMenuItem', iptrw);
20168:     XPManifest1', 'TXPManifest', iptrw);
20169:     SearchNext1', 'TMenuItem', iptrw);
20170:     Replace1', 'TMenuItem', iptrw);
20171:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
20172:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
20173:     ShowInclude1', 'TMenuItem', iptrw);
20174:     SynEditPrint1', 'TSynEditPrint', iptrw);
20175:     Printout1', 'TMenuItem', iptrw);
20176:     mnPrintColors1', 'TMenuItem', iptrw);
20177:     dlgFilePrint', 'TPrintDialog', iptrw);
20178:     dlgPrintFont1', 'TFontDialog', iptrw);
20179:     mnuPrintFont1', 'TMenuItem', iptrw);
20180:     Include1', 'TMenuItem', iptrw);
20181:     CodeCompletionList1', 'TMenuItem', iptrw);
20182:     IncludeList1', 'TMenuItem', iptrw);
20183:     ImageList1', 'TImageList', iptrw);
20184:     ImageList2', 'TImageList', iptrw);
20185:     CoolBar1', 'TCoolBar', iptrw);
20186:     ToolBar1', 'TToolBar', iptrw);
20187:     tbtnLoad', 'TToolButton', iptrw);
20188:     ToolButton2', 'TToolButton', iptrw);
20189:     tbtnFind', 'TToolButton', iptrw);
20190:     tbtnCompile', 'TToolButton', iptrw);
20191:     tbtnTrans', 'TToolButton', iptrw);
20192:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
20193:     toolbtnTutorial', 'TToolButton', iptrw);
20194:     tbtn6res', 'TToolButton', iptrw);
20195:     ToolButton5', 'TToolButton', iptrw);
20196:     ToolButton1', 'TToolButton', iptrw);
20197:     ToolButton3', 'TToolButton', iptrw);
20198:     statusBar1', 'TStatusBar', iptrw);
20199:     SaveOutput1', 'TMenuItem', iptrw);
20200:     ExportClipboard1', 'TMenuItem', iptrw);
20201:     Close1', 'TMenuItem', iptrw);
20202:     Manual1', 'TMenuItem', iptrw);
20203:     About2', 'TMenuItem', iptrw);
20204:     loadLastfile1', 'TMenuItem', iptrw);
20205:     imglogo', 'TImage', iptrw);
20206:     cedebug', 'TPSScriptDebugger', iptrw);
20207:     debugPopupMenu1', 'TPopupMenu', iptrw);
20208:     BreakPointMenu', 'TMenuItem', iptrw);
20209:     Decompile1', 'TMenuItem', iptrw);
20210:     StepInto1', 'TMenuItem', iptrw);
20211:     StepOut1', 'TMenuItem', iptrw);
20212:     Reset1', 'TMenuItem', iptrw);
20213:     DebugRun1', 'TMenuItem', iptrw);
20214:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
20215:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
20216:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
20217:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
20218:     tutorial4', 'TMenuItem', iptrw);
20219:     ExporttoClipboard1', 'TMenuItem', iptrw);
20220:     ImportfromClipboard1', 'TMenuItem', iptrw);
20221:     N4', 'TMenuItem', iptrw);
20222:     N5', 'TMenuItem', iptrw);
20223:     N6', 'TMenuItem', iptrw);
20224:     ImportfromClipboard2', 'TMenuItem', iptrw);
20225:     tutorial1', 'TMenuItem', iptrw);
20226:     N7', 'TMenuItem', iptrw);
20227:     ShowSpecChars1', 'TMenuItem', iptrw);
20228:     OpenDirectory1', 'TMenuItem', iptrw);
20229:     procMess', 'TMenuItem', iptrw);
20230:     tbtnUseCase', 'TToolButton', iptrw);
20231:     ToolButton7', 'TToolButton', iptrw);
20232:     EditFont1', 'TMenuItem', iptrw);
20233:     UseCase1', 'TMenuItem', iptrw);
20234:     tutorial21', 'TMenuItem', iptrw);
20235:     OpenUseCase1', 'TMenuItem', iptrw);
20236:     PSImport_DB1', 'TPSImport_DB', iptrw);
20237:     tutorial31', 'TMenuItem', iptrw);
20238:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
20239:     HTMLSyntax1', 'TMenuItem', iptrw);
20240:     ShowInterfaces1', 'TMenuItem', iptrw);
20241:     Tutorial5', 'TMenuItem', iptrw);
20242:     AllFunctionsList1', 'TMenuItem', iptrw);
20243:     ShowLastException1', 'TMenuItem', iptrw);
20244:     PlayMP31', 'TMenuItem', iptrw);
20245:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
20246:     texSyntax1', 'TMenuItem', iptrw);
20247:     N8', 'TMenuItem', iptrw);
20248:     GetEMails1', 'TMenuItem', iptrw);
20249:     SynCppSyn1', 'TSynCppSyn', iptrw);
20250:     CSyntax1', 'TMenuItem', iptrw);
```

```
20251:     Tutorial6', 'TMenuItem', iptrw);
20252:     New1', 'TMenuItem', iptrw);
20253:     AllObjectsList1', 'TMenuItem', iptrw);
20254:     LoadBytecode1', 'TMenuItem', iptrw);
20255:     CipherFile1', 'TMenuItem', iptrw);
20256:     N9', 'TMenuItem', iptrw);
20257:     N10', 'TMenuItem', iptrw);
20258:     Tutorial11', 'TMenuItem', iptrw);
20259:     Tutorial71', 'TMenuItem', iptrw);
20260:     UpdateService1', 'TMenuItem', iptrw);
20261:     PascalSchool1', 'TMenuItem', iptrw);
20262:     Tutorial81', 'TMenuItem', iptrw);
20263:     DelphiSite1', 'TMenuItem', iptrw);
20264:     Output1', 'TMenuItem', iptrw);
20265:     TerminalStyle1', 'TMenuItem', iptrw);
20266:     ReadOnly1', 'TMenuItem', iptrw);
20267:     ShellStyle1', 'TMenuItem', iptrw);
20268:     BigScreen1', 'TMenuItem', iptrw);
20269:     Tutorial91', 'TMenuItem', iptrw);
20270:     SaveOutput2', 'TMenuItem', iptrw);
20271:     N11', 'TMenuItem', iptrw);
20272:     SaveScreenshot', 'TMenuItem', iptrw);
20273:     Tutorial101', 'TMenuItem', iptrw);
20274:     SQLSyntax1', 'TMenuItem', iptrw);
20275:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
20276:     Console1', 'TMenuItem', iptrw);
20277:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
20278:     XMLSyntax1', 'TMenuItem', iptrw);
20279:     ComponentCount1', 'TMenuItem', iptrw);
20280:     NewInstance1', 'TMenuItem', iptrw);
20281:     toolbtnTutorial', 'TToolButton', iptrw);
20282:     Memory1', 'TMenuItem', iptrw);
20283:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
20284:     JavaSyntax1', 'TMenuItem', iptrw);
20285:     SyntaxCheck1', 'TMenuItem', iptrw);
20286:     Tutorial10Statistics1', 'TMenuItem', iptrw);
20287:     ScriptExplorer1', 'TMenuItem', iptrw);
20288:     FormOutput1', 'TMenuItem', iptrw);
20289:     ArduinoDump1', 'TMenuItem', iptrw);
20290:     AndroidDump1', 'TMenuItem', iptrw);
20291:     GotoEnd1', 'TMenuItem', iptrw);
20292:     AllResourceList1', 'TMenuItem', iptrw);
20293:     ToolButton4', 'TToolButton', iptrw);
20294:     tbtn6res', 'TToolButton', iptrw);
20295:     Tutorial11Forms1', 'TMenuItem', iptrw);
20296:     Tutorial12SQL1', 'TMenuItem', iptrw);
20297:     ResourceExplore1', 'TMenuItem', iptrw);
20298:     Info1', 'TMenuItem', iptrw);
20299:     N12', 'TMenuItem', iptrw);
20300:     CryptoBox1', 'TMenuItem', iptrw);
20301:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
20302:     CipherFile2', 'TMenuItem', iptrw);
20303:     N13', 'TMenuItem', iptrw);
20304:     ModulesCount1', 'TMenuItem', iptrw);
20305:     AddOns2', 'TMenuItem', iptrw);
20306:     N4GewinntGame1', 'TMenuItem', iptrw);
20307:     DocuforAddOns1', 'TMenuItem', iptrw);
20308:     Tutorial14Async1', 'TMenuItem', iptrw);
20309:     Lessons15Review1', 'TMenuItem', iptrw);
20310:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
20311:     PHPSyntax1', 'TMenuItem', iptrw);
20312:     Breakpoint1', 'TMenuItem', iptrw);
20313:     SerialRS2321', 'TMenuItem', iptrw);
20314:     N14', 'TMenuItem', iptrw);
20315:     SynCSSyn1', 'TSynCSSyn', iptrw);
20316:     CSyntax2', 'TMenuItem', iptrw);
20317:     Calculator1', 'TMenuItem', iptrw);
20318:     tbtnSerial', 'TToolButton', iptrw);
20319:     ToolButton8', 'TToolButton', iptrw);
20320:     Tutorial151', 'TMenuItem', iptrw);
20321:     N15', 'TMenuItem', iptrw);
20322:     N16', 'TMenuItem', iptrw);
20323:     ControlBar1', 'TControlBar', iptrw);
20324:     ToolBar2', 'TToolBar', iptrw);
20325:     BtnOpen', 'TToolButton', iptrw);
20326:     BtnSave', 'TToolButton', iptrw);
20327:     BtnPrint', 'TToolButton', iptrw);
20328:     BtnColors', 'TToolButton', iptrw);
20329:     btnClassReport', 'TToolButton', iptrw);
20330:     BtnRotateRight', 'TToolButton', iptrw);
20331:     BtnFullSize', 'TToolButton', iptrw);
20332:     BtnFitToWindowSize', 'TToolButton', iptrw);
20333:     BtnZoomMinus', 'TToolButton', iptrw);
20334:     BtnZoomPlus', 'TToolButton', iptrw);
20335:     Panel1', 'TPanel', iptrw);
20336:     LabelBrettgroesse', 'TLabel', iptrw);
20337:     CB1SCList', 'TComboBox', iptrw);
20338:     ImageListNormal', 'TImageList', iptrw);
20339:     spbtnexplore', 'TSpeedButton', iptrw);
```

```
20340:     spbtnexample', 'TSpeedButton', iptrw);
20341:     spbsaveas', 'TSpeedButton', iptrw);
20342:     imglogobox', 'TImage', iptrw);
20343:     EnlargeFont1', 'TMenuItem', iptrw);
20344:     EnlargeFont2', 'TMenuItem', iptrw);
20345:     ShrinkFont1', 'TMenuItem', iptrw);
20346:     ThreadDemo1', 'TMenuItem', iptrw);
20347:     HEXEditor1', 'TMenuItem', iptrw);
20348:     HEXView1', 'TMenuItem', iptrw);
20349:     HEXInspect1', 'TMenuItem', iptrw);
20350:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
20351:     ExporttoHTML1', 'TMenuItem', iptrw);
20352:     ClassCount1', 'TMenuItem', iptrw);
20353:     HTMLOutput1', 'TMenuItem', iptrw);
20354:     HEXEditor2', 'TMenuItem', iptrw);
20355:     Minesweeper1', 'TMenuItem', iptrw);
20356:     N17', 'TMenuItem', iptrw);
20357:     PicturePuzzle1', 'TMenuItem', iptrw);
20358:     sbvclhelp', 'TSpeedButton', iptrw);
20359:     DependencyWalker1', 'TMenuItem', iptrw);
20360:     WebScanner1', 'TMenuItem', iptrw);
20361:     View1', 'TMenuItem', iptrw);
20362:     mnToolbar1', 'TMenuItem', iptrw);
20363:     mnStatusbar2', 'TMenuItem', iptrw);
20364:     mnConsole2', 'TMenuItem', iptrw);
20365:     mnCoolbar2', 'TMenuItem', iptrw);
20366:     mnSplitter2', 'TMenuItem', iptrw);
20367:     WebServer1', 'TMenuItem', iptrw);
20368:     Tutorial17Server1', 'TMenuItem', iptrw);
20369:     Tutorial18Arduino1', 'TMenuItem', iptrw);
20370:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
20371:     PerlSyntax1', 'TMenuItem', iptrw);
20372:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
20373:     PythonSyntax1', 'TMenuItem', iptrw);
20374:     DMathLibrary1', 'TMenuItem', iptrw);
20375:     IntfNavigator1', 'TMenuItem', iptrw);
20376:     EnlargeFontConsole1', 'TMenuItem', iptrw);
20377:     ShrinkFontConsole1', 'TMenuItem', iptrw);
20378:     SetInterfaceList1', 'TMenuItem', iptrw);
20379:     popintfList', 'TPopupMenu', iptrw);
20380:     intfAdd1', 'TMenuItem', iptrw);
20381:     intfDelete1', 'TMenuItem', iptrw);
20382:     intfRefactor1', 'TMenuItem', iptrw);
20383:     Defactor1', 'TMenuItem', iptrw);
20384:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
20385:     Tutorial20Regex', 'TMenuItem', iptrw);
20386:     N18', 'TMenuItem', iptrw);
20387:     ManualE1', 'TMenuItem', iptrw);
20388:     FullTextFinder1', 'TMenuItem', iptrw);
20389:     Move1', 'TMenuItem', iptrw);
20390:     FractalDemo1', 'TMenuItem', iptrw);
20391:     Tutorial21Android1', 'TMenuItem', iptrw);
20392:     Tutorial0Function1', 'TMenuItem', iptrw);
20393:     SimuLogBox1', 'TMenuItem', iptrw);
20394:     OpenExamples1', 'TMenuItem', iptrw);
20395:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
20396:     JavaScriptSyntax1', 'TMenuItem', iptrw);
20397:     Halt1', 'TMenuItem', iptrw);
20398:     CodeSearch1', 'TMenuItem', iptrw);
20399:     SynRubySyn1', 'TSynRubySyn', iptrw);
20400:     RubySyntax1', 'TMenuItem', iptrw);
20401:     Undo1', 'TMenuItem', iptrw);
20402:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
20403:     LinuxShellScript1', 'TMenuItem', iptrw);
20404:     Rename1', 'TMenuItem', iptrw);
20405:     spdcodesearch', 'TSpeedButton', iptrw);
20406:     Preview1', 'TMenuItem', iptrw);
20407:     Tutorial22Services1', 'TMenuItem', iptrw);
20408:     Tutorial23RealTime1', 'TMenuItem', iptrw);
20409:     Configuration1', 'TMenuItem', iptrw);
20410:     MP3Player1', 'TMenuItem', iptrw);
20411:     DLLSpy1', 'TMenuItem', iptrw);
20412:     SynURIOpener1', 'TSynURIOpener', iptrw);
20413:     SynURISyn1', 'TSynURISyn', iptrw);
20414:     URILinksClicks1', 'TMenuItem', iptrw);
20415:     EditReplace1', 'TMenuItem', iptrw);
20416:     GotoLine1', 'TMenuItem', iptrw);
20417:     ActiveLineColor1', 'TMenuItem', iptrw);
20418:     ConfigFile1', 'TMenuItem', iptrw);
20419:     Sort1Intflist', 'TMenuItem', iptrw);
20420:     Redo1', 'TMenuItem', iptrw);
20421:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
20422:     Tutorial25Configuration1', 'TMenuItem', iptrw);
20423:     IndentSelection1', 'TMenuItem', iptrw);
20424:     UnindentSection1', 'TMenuItem', iptrw);
20425:     SkyStyle1', 'TMenuItem', iptrw);
20426:     N19', 'TMenuItem', iptrw);
20427:     CountWords1', 'TMenuItem', iptrw);
20428:     imbookmarkimages', 'TImageList', iptrw);
```

```
20429:     Bookmark11', 'TMenuItem', iptrw);
20430:     N20', 'TMenuItem', iptrw);
20431:     Bookmark21', 'TMenuItem', iptrw);
20432:     Bookmark31', 'TMenuItem', iptrw);
20433:     Bookmark41', 'TMenuItem', iptrw);
20434:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
20435:
20436:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
20437:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
20438:     Procedure PSScriptCompile( Sender : TPSScript)
20439:     Procedure Compile1Click( Sender : TObject)
20440:     Procedure PSScriptExecute( Sender : TPSScript)
20441:     Procedure open1Click( Sender : TObject)
20442:     Procedure Save2Click( Sender : TObject)
20443:     Procedure Savebefore1Click( Sender : TObject)
20444:     Procedure Largefont1Click( Sender : TObject)
20445:     Procedure FormActivate( Sender : TObject)
20446:     Procedure SBytecode1Click( Sender : TObject)
20447:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
20448:     Procedure Saveas3Click( Sender : TObject)
20449:     Procedure Clear1Click( Sender : TObject)
20450:     Procedure Slinenumbers1Click( Sender : TObject)
20451:     Procedure About1Click( Sender : TObject)
20452:     Procedure Search1Click( Sender : TObject)
20453:     Procedure FormCreate( Sender : TObject)
20454:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
20455:                                                     var Action : TSynReplaceAction)
20456:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
20457:     Procedure WordWrap1Click( Sender : TObject)
20458:     Procedure SearchNext1Click( Sender : TObject)
20459:     Procedure Replace1Click( Sender : TObject)
20460:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
20461:     Procedure ShowInclude1Click( Sender : TObject)
20462:     Procedure Printout1Click( Sender : TObject)
20463:     Procedure mnuPrintFont1Click( Sender : TObject)
20464:     Procedure Include1Click( Sender : TObject)
20465:     Procedure FormDestroy( Sender : TObject)
20466:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
20467:     Procedure UpdateView1Click( Sender : TObject)
20468:     Procedure CodeCompletionList1Click( Sender : TObject)
20469:     Procedure SaveOutput1Click( Sender : TObject)
20470:     Procedure ExportClipboard1Click( Sender : TObject)
20471:     Procedure Close1Click( Sender : TObject)
20472:     Procedure Manual1Click( Sender : TObject)
20473:     Procedure LoadLastFile1Click( Sender : TObject)
20474:     Procedure Memo1Change( Sender : TObject)
20475:     Procedure Decompile1Click( Sender : TObject)
20476:     Procedure StepInto1Click( Sender : TObject)
20477:     Procedure StepOut1Click( Sender : TObject)
20478:     Procedure Reset1Click( Sender : TObject)
20479:     Procedure cedebugAfterExecute( Sender : TPSScript)
20480:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
20481:     Procedure cedebugCompile( Sender : TPSScript)
20482:     Procedure cedebugExecute( Sender : TPSScript)
20483:     Procedure cedebugIdle( Sender : TObject)
20484:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
20485:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
20486:     Procedure BreakPointMenuClick( Sender : TObject)
20487:     Procedure DebugRun1Click( Sender : TObject)
20488:     Procedure tutorial4Click( Sender : TObject)
20489:     Procedure ImportfromClipboard1Click( Sender : TObject)
20490:     Procedure ImportfromClipboard2Click( Sender : TObject)
20491:     Procedure tutorial1Click( Sender : TObject)
20492:     Procedure ShowSpecChars1Click( Sender : TObject)
20493:     Procedure StatusBar1DblClick( Sender : TObject)
20494:     Procedure PSScriptLine( Sender : TObject)
20495:     Procedure OpenDirectory1Click( Sender : TObject)
20496:     Procedure procMessClick( Sender : TObject)
20497:     Procedure tbtnUseCaseClick( Sender : TObject)
20498:     Procedure EditFont1Click( Sender : TObject)
20499:     Procedure tutorial21Click( Sender : TObject)
20500:     Procedure tutorial31Click( Sender : TObject)
20501:     Procedure HTMLSyntax1Click( Sender : TObject)
20502:     Procedure ShowInterfaces1Click( Sender : TObject)
20503:     Procedure Tutorial5Click( Sender : TObject)
20504:     Procedure ShowLastException1Click( Sender : TObject)
20505:     Procedure PlayMP31Click( Sender : TObject)
20506:     Procedure AllFunctionsList1Click( Sender : TObject)
20507:     Procedure texSyntax1Click( Sender : TObject)
20508:     Procedure GetEMails1Click( Sender : TObject)
20509:     procedure DelphiSite1Click(Sender: TObject);
20510:     procedure TerminalStyle1Click(Sender: TObject);
20511:     procedure ReadOnly1Click(Sender: TObject); -->maxform1.memo2.readonly:= false;
20512:     procedure ShellStyle1Click(Sender: TObject);
20513:     procedure Console1Click(Sender: TObject);     //3.2
20514:     procedure BigScreen1Click(Sender: TObject);
20515:     procedure Tutorial91Click(Sender: TObject);
20516:     procedure SaveScreenshotClick(Sender: TObject);
20517:     procedure Tutorial101Click(Sender: TObject);
```

```
20518:     procedure SQLSyntax1Click(Sender: TObject);
20519:     procedure XMLSyntax1Click(Sender: TObject);
20520:     procedure ComponentCount1Click(Sender: TObject);
20521:     procedure NewInstance1Click(Sender: TObject);
20522:     procedure CSyntax1Click(Sender: TObject);
20523:     procedure Tutorial6Click(Sender: TObject);
20524:     procedure New1Click(Sender: TObject);
20525:     procedure AllObjectsList1Click(Sender: TObject);
20526:     procedure LoadBytecode1Click(Sender: TObject);
20527:     procedure CipherFile1Click(Sender: TObject);  //V3.5
20528:     procedure NewInstance1Click(Sender: TObject);
20529:     procedure toolbtnTutorialClick(Sender: TObject);
20530:     procedure Memory1Click(Sender: TObject);
20531:     procedure JavaSyntax1Click(Sender: TObject);
20532:     procedure SyntaxCheck1Click(Sender: TObject);
20533:     procedure ScriptExplorer1Click(Sender: TObject);
20534:     procedure FormOutput1Click(Sender: TObject);  //V3.6
20535:     procedure GotoEnd1Click(Sender: TObject);
20536:     procedure AllResourceList1Click(Sender: TObject);
20537:     procedure tbtn6resClick(Sender: TObject);   //V3.7
20538:     procedure Info1Click(Sender: TObject);
20539:     procedure Tutorial10Statistics1Click(Sender: TObject);
20540:     procedure Tutorial11Forms1Click(Sender: TObject);
20541:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
20542:     procedure ResourceExplore1Click(Sender: TObject);
20543:     procedure Info1Click(Sender: TObject);
20544:     procedure CryptoBox1Click(Sender: TObject);
20545:     procedure ModulesCount1Click(Sender: TObject);
20546:     procedure N4GewinntGame1Click(Sender: TObject);
20547:     procedure PHPSyntax1Click(Sender: TObject);
20548:     procedure SerialRS2321Click(Sender: TObject);
20549:     procedure CSyntax2Click(Sender: TObject);
20550:     procedure Calculator1Click(Sender: TObject);
20551:     procedure Tutorial13Ciphering1Click(Sender: TObject);
20552:     procedure Tutorial14Async1Click(Sender: TObject);
20553:     procedure PHPSyntax1Click(Sender: TObject);
20554:     procedure BtnZoomPlusClick(Sender: TObject);
20555:     procedure BtnZoomMinusClick(Sender: TObject);
20556:     procedure btnClassReportClick(Sender: TObject);
20557:     procedure ThreadDemo1Click(Sender: TObject);
20558:     procedure HEXView1Click(Sender: TObject);
20559:     procedure ExporttoHTML1Click(Sender: TObject);
20560:     procedure Minesweeper1Click(Sender: TObject);
20561:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
20562:     procedure sbvclhelpClick(Sender: TObject);
20563:     procedure DependencyWalker1Click(Sender: TObject);
20564:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
20565:     procedure WebScanner1Click(Sender: TObject);
20566:     procedure mnToolbar1Click(Sender: TObject);
20567:     procedure mnStatusbar2Click(Sender: TObject);
20568:     procedure mnConsole2Click(Sender: TObject);
20569:     procedure mnCoolbar2Click(Sender: TObject);
20570:     procedure mnSplitter2Click(Sender: TObject);
20571:     procedure WebServer1Click(Sender: TObject);
20572:     procedure PerlSyntax1Click(Sender: TObject);
20573:     procedure PythonSyntax1Click(Sender: TObject);
20574:     procedure DMathLibrary1Click(Sender: TObject);
20575:     procedure IntfNavigator1Click(Sender: TObject);
20576:     procedure FullTextFinder1Click(Sender: TObject);
20577:     function AppName: string;
20578:     function ScriptName: string;
20579:     function LastName: string;
20580:     procedure FractalDemo1Click(Sender: TObject);
20581:     procedure SimuLogBox1Click(Sender: TObject);
20582:     procedure OpenExamples1Click(Sender: TObject);
20583:     procedure Halt1Click(Sender: TObject);
20584:     procedure Stop;
20585:     procedure CodeSearch1Click(Sender: TObject);
20586:     procedure RubySyntax1Click(Sender: TObject);
20587:     procedure Undo1Click(Sender: TObject);
20588:     procedure LinuxShellScript1Click(Sender: TObject);
20589:     procedure WebScannerDirect(urls: string);
20590:     procedure WebScanner(urls: string);
20591:     procedure LoadInterfaceList2;
20592:     procedure DLLSpy1Click(Sender: TObject);
20593:     procedure Memo1DblClick(Sender: TObject);
20594:     procedure URILinksClicks1Click(Sender: TObject);
20595:     procedure GotoLine1Click(Sender: TObject);
20596:     procedure ConfigFile1Click(Sender: TObject);
20597:     Procedure Sort1IntflistClick( Sender : TObject)
20598:     Procedure Redo1Click( Sender : TObject)
20599:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
20600:     Procedure IndentSelection1Click( Sender : TObject)
20601:     Procedure UnindentSection1Click( Sender : TObject)
20602:     Procedure SkyStyle1Click( Sender : TObject)
20603:     Procedure CountWords1Click( Sender : TObject)
20604:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
20605:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
20606:     Procedure Bookmark11Click( Sender : TObject)
```

```
20607:     Procedure Bookmark21Click( Sender : TObject)
20608:     Procedure Bookmark31Click( Sender : TObject)
20609:     Procedure Bookmark41Click( Sender : TObject)
20610:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
20611:     'STATMemoryReport', 'boolean', iptrw);
20612:     'IPPort', 'integer', iptrw);
20613:     'COMPort', 'integer', iptrw);
20614:     'lbintflist', 'TListBox', iptrw);
20615:     Function GetStatChange : boolean
20616:     Procedure SetStatChange( vstat : boolean)
20617:     Function GetActFileName : string
20618:     Procedure SetActFileName( vname : string)
20619:     Function GetLastFileName : string
20620:     Procedure SetLastFileName( vname : string)
20621:     Procedure WebScannerDirect( urls : string)
20622:     Procedure LoadInterfaceList2
20623:     Function GetStatExecuteShell : boolean
20624:     Procedure DoEditorExecuteCommand( EditorCommand : word)
20625:     function GetActiveLineColor: TColor
20626:     procedure SetActiveLineColor(acolor: TColor)
20627:     procedure ScriptListbox1Click(Sender: TObject);
20628:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
20629:     procedure EnlargeGutter1Click(Sender: TObject);
20630:     procedure Tetris1Click(Sender: TObject);
20631:     procedure ToDoList1Click(Sender: TObject);
20632:     procedure ProcessList1Click(Sender: TObject);
20633:     procedure MetricReport1Click(Sender: TObject);
20634:     procedure ProcessList1Click(Sender: TObject);
20635:     procedure TCPSockets1Click(Sender: TObject);
20636:     procedure ConfigUpdate1Click(Sender: TObject);
20637:     procedure ADOWorkbench1Click(Sender: TObject);
20638:     procedure SocketServer1Click(Sender: TObject);
20639:     procedure FormDemo1Click(Sender: TObject);
20640:     procedure Richedit1Click(Sender: TObject);
20641:     procedure SimpleBrowser1Click(Sender: TObject);
20642:     procedure DOSShell1Click(Sender: TObject);
20643:     procedure SynExport1Click(Sender: TObject);
20644:     procedure ExporttoRTF1Click(Sender: TObject);
20645:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
20646:     procedure SOAPTester1Click(Sender: TObject);
20647:     procedure Sniffer1Click(Sender: TObject);
20648:     procedure AutoDetectSyntax1Click(Sender: TObject);
20649:     procedure FPlot1Click(Sender: TObject);
20650:     procedure PasStyle1Click(Sender: TObject);
20651:     procedure Tutorial183RGBLED1Click(Sender: TObject);
20652:     procedure Reversi1Click(Sender: TObject);
20653:     procedure ManualmaXbox1Click(Sender: TObject);
20654:     procedure BlaisePascalMagazine1Click(Sender: TObject);
20655:     procedure AddToDo1Click(Sender: TObject);
20656:     procedure CreateGUID1Click(Sender: TObject);
20657:     procedure Tutorial27XML1Click(Sender: TObject);
20658:     procedure CreateDLLStub1Click(Sender: TObject);
20659:     procedure Tutorial28DLL1Click(Sender: TObject);');
20660:     procedure ResetKeyPressed;');
20661:     procedure KeyPressedFalse;
20662:     procedure FileChanges1Click(Sender: TObject);');
20663:     procedure OpenGLTry1Click(Sender: TObject);');
20664:     procedure AllUnitList1Click(Sender: TObject);');
20665:     procedure Tutorial29UMLClick(Sender: TObject);
20666:     procedure CreateHeader1Click(Sender: TObject);
20667:     procedure Oscilloscope1Click(Sender: TObject);');
20668:     procedure Tutorial30WOT1Click(Sender: TObject);');
20669:     procedure GetWebScript1Click(Sender: TObject);');
20670:     procedure Checkers1Click(Sender: TObject);');
20671:     procedure TaskMgr1Click(Sender: TObject);');
20672:     procedure WebCam1Click(Sender: TObject);');
20673:     procedure Tutorial31Closure1Click(Sender: TObject);');
20674:     procedure GEOMapView1Click(Sender: TObject);');
20675:
20676:
20677: //----------------------------------------------------------------------------
20678: //*************mX4 Editor SynEdit Tools API ********************************
20679: //----------------------------------------------------------------------------
20680: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
20681: begin
20682:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
20683:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
20684:     Constructor Create( AOwner : TComponent)
20685:     SelStart', 'Integer', iptrw);
20686:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
20687:     Procedure UpdateCaret
20688:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
20689:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
20690:     Procedure BeginUndoBlock
20691:     Procedure BeginUpdate
20692:     Function CaretInView : Boolean
20693:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
20694:     Procedure Clear
20695:     Procedure ClearAll
```

```
20696:     Procedure ClearBookMark( BookMark : Integer)
20697:     Procedure ClearSelection
20698:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
20699:     Procedure ClearUndo
20700:     Procedure CopyToClipboard
20701:     Procedure CutToClipboard
20702:     Procedure DoCopyToClipboard( const SText : string)
20703:     Procedure EndUndoBlock
20704:     Procedure EndUpdate
20705:     Procedure EnsureCursorPosVisible
20706:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
20707:     Procedure FindMatchingBracket
20708:     Function GetMatchingBracket : TBufferCoord
20709:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
20710:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
20711:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
20712:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
20713:               : TSynHighlighterAttributes) : boolean
20714:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
20715:              var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
20716:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
20717:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
20718:     Procedure GotoBookMark( BookMark : Integer)
20719:     Procedure GotoLineAndCenter( ALine : Integer)
20720:     Function IdentChars : TSynIdentChars
20721:     Procedure InvalidateGutter
20722:     Procedure InvalidateGutterLine( aLine : integer)
20723:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
20724:     Procedure InvalidateLine( Line : integer)
20725:     Procedure InvalidateLines( FirstLine, LastLine : integer)
20726:     Procedure InvalidateSelection
20727:     Function IsBookmark( BookMark : integer) : boolean
20728:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
20729:     Procedure LockUndo
20730:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
20731:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
20732:     Function LineToRow( aLine : integer) : integer
20733:     Function RowToLine( aRow : integer) : integer
20734:     Function NextWordPos : TBufferCoord
20735:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
20736:     Procedure PasteFromClipboard
20737:     Function WordStart : TBufferCoord
20738:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
20739:     Function WordEnd : TBufferCoord
20740:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
20741:     Function PrevWordPos : TBufferCoord
20742:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
20743:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
20744:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
20745:     Procedure Redo
20746:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
20747:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
20748:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
20749:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
20750:     Procedure SelectAll
20751:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
20752:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
20753:     Procedure SetDefaultKeystrokes
20754:     Procedure SetSelWord
20755:     Procedure SetWordBlock( Value : TBufferCoord)
20756:     Procedure Undo
20757:     Procedure UnlockUndo
20758:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
20759:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
20760:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
20761:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
20762:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
20763:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
20764:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
20765:     Procedure AddFocusControl( aControl : TWinControl)
20766:     Procedure RemoveFocusControl( aControl : TWinControl)
20767:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
20768:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
20769:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
20770:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
20771:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
20772:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
20773:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
20774:     Procedure RemoveLinesPointer
20775:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
20776:     Procedure UnHookTextBuffer
20777:     BlockBegin', 'TBufferCoord', iptrw);
20778:     BlockEnd', 'TBufferCoord', iptrw);
20779:     CanPaste', 'Boolean', iptr);
20780:     CanRedo', 'boolean', iptr);
20781:     CanUndo', 'boolean', iptr);
20782:     CaretX', 'Integer', iptrw);
20783:     CaretY', 'Integer', iptrw);
20784:     CaretXY', 'TBufferCoord', iptrw);
```

```
20785:     ActiveLineColor', 'TColor', iptrw);
20786:     DisplayX', 'Integer', iptr);
20787:     DisplayY', 'Integer', iptr);
20788:     DisplayXY', 'TDisplayCoord', iptr);
20789:     DisplayLineCount', 'integer', iptr);
20790:     CharsInWindow', 'Integer', iptr);
20791:     CharWidth', 'integer', iptr);
20792:     Font', 'TFont', iptrw);
20793:     GutterWidth', 'Integer', iptr);
20794:     Highlighter', 'TSynCustomHighlighter', iptrw);
20795:     LeftChar', 'Integer', iptrw);
20796:     LineHeight', 'integer', iptr);
20797:     LinesInWindow', 'Integer', iptr);
20798:     LineText', 'string', iptrw);
20799:     Lines', 'TStrings', iptrw);
20800:     Marks', 'TSynEditMarkList', iptr);
20801:     MaxScrollWidth', 'integer', iptrw);
20802:     Modified', 'Boolean', iptrw);
20803:     PaintLock', 'Integer', iptr);
20804:     ReadOnly', 'Boolean', iptrw);
20805:     SearchEngine', 'TSynEditSearchCustom', iptrw);
20806:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
20807:     SelTabBlock', 'Boolean', iptr);
20808:     SelTabLine', 'Boolean', iptr);
20809:     SelText', 'string', iptrw);
20810:     StateFlags', 'TSynStateFlags', iptr);
20811:     Text', 'string', iptrw);
20812:     TopLine', 'Integer', iptrw);
20813:     WordAtCursor', 'string', iptr);
20814:     WordAtMouse', 'string', iptr);
20815:     UndoList', 'TSynEditUndoList', iptr);
20816:     RedoList', 'TSynEditUndoList', iptr);
20817:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
20818:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
20819:     BorderStyle', 'TSynBorderStyle', iptrw);
20820:     ExtraLineSpacing', 'integer', iptrw);
20821:     Gutter', 'TSynGutter', iptrw);
20822:     HideSelection', 'boolean', iptrw);
20823:     InsertCaret', 'TSynEditCaretType', iptrw);
20824:     InsertMode', 'boolean', iptrw);
20825:     IsScrolling', 'Boolean', iptr);
20826:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
20827:     MaxUndo', 'Integer', iptrw);
20828:     Options', 'TSynEditorOptions', iptrw);
20829:     OverwriteCaret', 'TSynEditCaretType', iptrw);
20830:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
20831:     ScrollHintColor', 'TColor', iptrw);
20832:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
20833:     ScrollBars', 'TScrollStyle', iptrw);
20834:     SelectedColor', 'TSynSelectedColor', iptrw);
20835:     SelectionMode', 'TSynSelectionMode', iptrw);
20836:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
20837:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
20838:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
20839:     WordWrapGlyph', 'TSynGlyph', iptrw);
20840:     OnChange', 'TNotifyEvent', iptrw);
20841:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
20842:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
20843:     OnContextHelp', 'TContextHelpEvent', iptrw);
20844:     OnDropFiles', 'TDropFilesEvent', iptrw);
20845:     OnGutterClick', 'TGutterClickEvent', iptrw);
20846:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
20847:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
20848:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
20849:     OnPaint', 'TPaintEvent', iptrw);
20850:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
20851:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
20852:     OnReplaceText', 'TReplaceTextEvent', iptrw);
20853:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
20854:     OnStatusChange', 'TStatusChangeEvent', iptrw);
20855:     OnPaintTransient', 'TPaintTransient', iptrw);
20856:     OnScroll', 'TScrollEvent', iptrw);
20857:   end;
20858:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
20859:  Function GetPlaceableHighlighters : TSynHighlighterList
20860:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
20861:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
20862:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
20863:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
20864:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
20865:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
20866:  Function ConvertCodeStringToExtended( AString : String) : String
20867:  Function ConvertExtendedToCodeString( AString : String) : String
20868:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
20869:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
20870:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
20871:
20872:   TSynEditorOption = (
20873:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
```

```
20874:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
20875:                                // preceding line
20876:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
20877:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
20878:                                //direction any more
20879:     eoDragDropEditing,         //Allows to select a textblock and drag it in document to another location
20880:     eoDropFiles,               //Allows the editor accept OLE file drops
20881:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
20882:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
20883:     eoGroupUndo,               //When undoing/redoing actions,handle all cont.changes same kind in onecall
20884:                                //instead undoing/redoing each command separately
20885:     eoHalfPageScroll,          //By scrolling with page-up/page-down commands,only scroll half page attime
20886:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
20887:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
20888:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
20889:     eoNoCaret,                 //Makes it so the caret is never visible
20890:     eoNoSelection,             //Disables selecting text
20891:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
20892:     eoScrollByOneLess,         //Forces scrolling to be one less
20893:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
20894:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
20895:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
20896:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
20897:     eoShowSpecialChars,        //Shows the special Characters
20898:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
20899:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
20900:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
20901:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
20902:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
20903:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
20904:
20905:     *********************************Important Editor Short Cuts*****************************);
20906:     Double click to select a word and count words with highlightning.
20907:     Triple click to select a line.
20908:     CTRL+SHIFT+click to extend a selection.
20909:     Drag with the ALT key down to select columns of text !!!
20910:     Drag and drop is supported.
20911:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
20912:     Type CTRL+A to select all.
20913:     Type CTRL+N to set a new line.
20914:     Type CTRL+T to delete a line or token. //Tokenizer
20915:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
20916:     Type CTRL+Shift+T to add ToDo in line and list.
20917:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
20918:     Type CTRL[0..9] to jump or get to bookmarks.
20919:     Type Home to position cursor at beginning of current line and End to position it at end of line.
20920:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
20921:     Page Up and Page Down work as expected.
20922:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
20923:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
20924:
20925: {$ Short Key Positions Ctrl<A-Z>: }
20926: def
20927:    <A> Select All
20928:    <B> Count Words
20929:    <C> Copy
20930:    <D> Internet Start
20931:    <E> Script List
20932:    <F> Find
20933:    <G> Goto
20934:    <H> Mark Line
20935:    <I> Interface List
20936:    <J> Code Completion
20937:    <K> Console
20938:    <L> Interface List Box
20939:    <M> Font Larger -
20940:    <N> New Line
20941:    <O> Open File
20942:    <P> Font Smaller +
20943:    <Q> Quit
20944:    <R> Replace
20945:    <S> Save!
20946:    <T> Delete Line
20947:    <U> Use Case Editor
20948:    <V> Paste
20949:    <W> URI Links
20950:    <X> Reserved for coding use internal
20951:    <Y> Delete Line
20952:    <Z> Undo
20953:
20954: ref
20955:    F1 Help
20956:    F2 Syntax Check
20957:    F3 Search Next
20958:    F4 New Instance
20959:    F5 Line Mark /Breakpoint
20960:    F6 Goto End
20961:    F7 Debug Step Into
20962:    F8 Debug Step Out
```

```
20963:    F9 Compile
20964:    F10 Menu
20965:    F11 Word Count Highlight
20966:    F12 Reserved for coding use internal
20967:
20968:    AddRegisteredVariable( it ,integer');  //for closure!!
20969:    AddRegisteredVariable( sr ,string');  //for closure
20970:    AddRegisteredVariable( bt ,boolean');  //for closure
20971:    AddRegisteredVariable( ft ,double');  //for closure
20972:    AddRegisteredVariable( srlist ,TStringlist');  //for closures
20973:
20974:  def ReservedWords: array[0..82] of string =
20975:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
20976:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
20977:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
20978:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
20979:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
20980:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
20981:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
20982:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
20983:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
20984:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
20985:      'public', 'published',def,ref,using,typedef ,memo1','memo2','doc','maxform1','it';
20986:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
20987: //-------------------------------------------------------------------------
20988: //*************End of mX4 Public  Tools API *********************************
20989: //-------------------------------------------------------------------------
20990:
20991: Amount of Functions: 13419
20992: Amount of Procedures: 8289
20993: Amount of Constructors: 1337
20994: Totals of Calls: 23045
20995: SHA1: Win 3.9.9.98 7C600FFC801FFF2AAEC83DD220202DDF5271C64D
20996:
20997:
20998: ************************************************************
20999:  Doc Short Manual with 50 Tips!
21000: ************************************************************
21001: - Install: just save your maxboxdef.ini before and then extract the zip file!
21002: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
21003: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
21004: - Menu: With <Crtl><F3> you can search for code on examples
21005: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
21006: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
21007: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
21008:
21009: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
21010: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
21011: - Context Menu: You can printout your scripts as a pdf-file or html-export
21012: - Context: You do have a context menu with the right mouse click
21013:
21014: - Menu: With the UseCase Editor you can convert graphic formats too.
21015: - Menu: On menu Options you find Addons as compiled scripts
21016: - IDE: You don't need a mouse to handle maXbox, use shortcuts
21017: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
21018: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
21019: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
21020:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
21021:
21022: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
21023: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
21024: - Code: If you code a loop till key-pressed use function: isKeyPressed;
21025: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
21026: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
21027: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
21028:          to delete and Click and mark to drag a bookmark
21029: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
21030: - IDE: A file info with system and script information you find in menu Program/Information
21031: - IDE: After change the config file in help you can update changes in menu Help/Config Update
21032: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
21033: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
21034: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
21035: - Editor: Set Bookmarks to check your work in app or code
21036: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
21037: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
21038: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
21039:
21040: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
21041: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
21042: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
21043: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
21044: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
21045: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
21046: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
21047: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
21048: - IDE menu /Help/Tools/ open the Task Manager
21049:
21050: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
21051: - Add on when no browser is available start /Options/Add ons/Easy Browser
```

```
21052: - Add on SOAP Tester with SOP POST File
21053: - Add on IP Protocol Sniffer with List View
21054: - Add on OpenGL mX Robot Demo for android
21055: - Add on Checkers Game
21056:   Add on Oscilloscope
21057:
21058: - Menu: Help/Tools as a Tool Section with DOS Opener
21059: - Menu Editor: export the code as RTF File
21060: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
21061: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
21062: - Context: Auto Detect of Syntax depending on file extension
21063: - Code: some Windows API function start with w in the name like wGetAtomName();
21064: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
21065: - IDE File Check with menu ..View/File Changes/...
21066: - Context: Create a Header with Create Header in Navigator List at right window
21067: - Code: use  SysErrorMessage to get a real Error Description, Ex.
21068:         RemoveDir('c:\NoSuchFolder') ;
21069:         writeln('System Error Message: '+ SysErrorMessage(GetLastError));
21070: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
21071:
21072: - using DLL example in maXbox: //function: {*****************************************}
21073:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
21074:                                  cb: DWORD): BOOL; //stdcall;;
21075:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
21076:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
21077:      External  'OpenProcess@kernel32.dll stdcall';
21078:
21079: GCC Compile Ex Script
21080: procedure TFormMain_btnCompileClick(Sender: TObject);
21081: begin
21082:   AProcess:= TProcess.Create(Nil);
21083:   try  AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"'
21084:       + ' -o "' + OpenDialog2.FileName + '"';
21085:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
21086:     AProcess.Execute;
21087:     Memo2.Lines.BeginUpdate;
21088:     Memo2.Lines.Clear;
21089:     Memo2.Lines.LoadFromStream(AProcess.Output);
21090:     Memo2.Lines.EndUpdate;
21091:   finally
21092:     AProcess.Free;
21093:   end;
21094: end;
21095:
21096: Stopwatch pattern
21097: Time1:= Time;
21098:   writeln(formatdatetime('"start:" hh:mm:ss:zzz',Time))
21099:   if initAndStartBoard then
21100:      writeln('Filesize: '+inttoStr(filesize(FILESAVE)));
21101:   writeln(formatDateTime('"stop:" hh:mm:ss:zzz',Time))
21102:   PrintF('%d %s',[Trunc((Time-Time1)*24),
21103:                   FormatDateTime('"h runtime:" nn:ss:zzz',Time-Time1)])
21104:
21105:    POST git-receive-pack (chunked)
21106: Pushing to https://github.com/maxkleiner/maXbox3.git
21107: To https://github.com/maxkleiner/maXbox3.git
21108:    f127d21..c6a98da  masterbox2 -> masterbox2
21109: updating local tracking ref 'refs/remotes/maXbox3Remote/masterbox2'
21110:
21111: History Shell Hell
21112: PCT Precompile Technology , mX4 ScriptStudio
21113: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
21114: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
21115: emax layers: system-package-component-unit-class-function-block
21116: new keywords def ref using maXCalcF
21117: UML: use case act class state seq pac comp dep - lib lab
21118: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
21119: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
21120: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
21121: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
21122: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
21123: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
21124: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
21125: SendBuffer, Color+Caption hack, ComboSet,SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
21126: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
21127: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
21128: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21129: Inno Install and Setup Routines
21130: Add 32 Units, Wav res, RoundTo, OpenOffice, Pipes, GSM2
21131: TFixedCriticalSection, XPlatform beta, GCC Command Pipe
21132: VfW (Video), FindFirst3, ResFiler, AssemblyCache, UnitTest
21133: 9 Color LED, LED Resources, Runtime LED, it + sr var , morse generator
21134: Add 5 Units, 1 Tutors, maXmap, OpenStreetView, MAPX
21135: Function Menu/View/GEO Map View, DownloadFile, wgetX, sensors
21136: StreamUtils, IDL Syntax, OpenStreetMap
21137:
21138:
21139: Ref:
21140: https://unibe-ch.academia.edu/MaxKleiner
```

```
21141: http://www.slideshare.net/maxkleiner1
21142: http://www.scribd.com/max_kleiner
21143: http://www.delphiforfun.org/Programs/Utilities/index.htm
21144: http://www.slideshare.net/maxkleiner1
21145: http://s3.amazonaws.com/PreviewLinks/22959.html
21146: http://www.softwareschule.ch/arduino_training.pdf
21147: http://www.jrsoftware.org/isinfo.php
21148: http://www.be-precision.com/products/precision-builder/express/
21149: http://www.blaisepascal.eu/
21150: http://www.delphibasics.co.uk/
21151: http://www.youtube.com/watch?v=av89HAbqAsI
21152: http://www.angelfire.com/hi5/delphizeus/modal.html
21153: https://github.com/dilshan/signalman/blob/master/SignalManTemplate.pas
21154: http://delphi.org/2014/01/every-android-api-for-delphi/
21155:
21156: UrlGoogleQrCode='http://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s&chl=%s';
21157: UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search.php?format=%s&json_callback
21158: =renderBasicSearchNarrative&q=%s';
21159: UrlMapQuestAPIReverse:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?format=
21160: %s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
21161:
21162:
21163: function OpenMap(const Data: string): boolean;
21164: var encURL: string;
21165: begin
21166:   encURL:= Format(UrlMapQuestAPICode2,['html',HTTPEncode(Data)]);
21167:   try
21168:    //HttpGet(EncodedURL, mapStream);   //WinInet
21169:    Result:= UrlDownloadToFile(Nil,PChar(encURL),PChar(Exepath+'openmapx.html'),0,Nil)= 0;
21170:    //OpenDoc(Exepath+'openmapx.html');
21171:    S_ShellExecute(Exepath+'openmapx.html','',seCmdOpen);
21172:   finally
21173:     encURL:= '';
21174:   end;
21175: end;
21176:
21177: procedure GetGEOMap(C_form,apath: string; const Data: string);
21178: var
21179:   encodedURL: string;
21180:   mapStream: TMemoryStream;
21181: begin
21182:   //encodedURL:= Format(UrlGoogleQrCode,[Width,Height, C_Level, HTTPEncode(Data)]);
21183:   encodedURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
21184:   mapStream:= TMemoryStream.create;
21185:   try
21186:     Wininet_HttpGet(EncodedURL, mapStream);   //WinInet
21187:     mapStream.Position:= 0;
21188:     mapStream.Savetofile(apath);
21189:     // OpenDoc(apath);
21190:     S_ShellExecute(apath,'',seCmdOpen);
21191:   finally
21192:     mapStream.Free;
21193:   end;
21194: end;
21195:
21196:
21197:
21198: ************************************************************
21199:  unit List asm internal end
21200: ************************************************************
21201: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
21202: 02 unit SIRegister_IdStrings                   //Indy Sockets
21203: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
21204: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
21205: 05 unit IFSI_WinForm1puzzle;                   //maXbox
21206: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
21207: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
21208: 08 unit RIRegister_MathMax_Routines(exec);     //Jedi & Delphi
21209: 09 unit RIRegister_IdGlobal_Routines(exec);    //Indy Sockets
21210: 10 unit RIRegister_SysUtils_Routines(Exec);    //Delphi
21211: 11 unit uPSI_IdTCPConnection;                  //Indy some functions
21212: 12 unit uPSCompiler.pas;                       //PS kernel functions
21213: 13 unit uPSI_DBCommon;                         //DB Common_Routines and Types
21214: 14 unit uPSI_Printers.pas                      //Delphi VCL
21215: 15 unit uPSI_MPlayer.pas                      //Delphi VCL
21216: 16 unit uPSC_comobj;                          //COM Functions
21217: 17 unit uPSI_Clipbrd;                         //Delphi VCL
21218: 18 unit Filectrl in IFSI_SysUtils_max;        //VCL Runtime
21219: 19 unit uPSI_SqlExpr;                         //DBX3
21220: 20 unit uPSI_ADODB;                           //ADODB
21221: 21 unit uPSI_StrHlpr;                         //String Helper Routines
21222: 22 unit uPSI_DateUtils;                       //Expansion to DateTimeLib
21223: 23 unit uPSI_FileUtils;                       //Expansion to Sys/File Utils
21224: 24 unit JUtils / gsUtils;                     //Jedi / Metabase
21225: 25 unit JvFunctions_max;                      //Jedi Functions
21226: 26 unit HTTPParser;                           //Delphi VCL
21227: 27 unit HTTPUtil;                            //Delphi VCL
21228: 28 unit uPSI_XMLUtil;                        //Delphi VCL
21229: 29 unit uPSI_SOAPHTTPClient;                  //Delphi VCL SOAP WebService V3.5
```

```
21230: 30 unit uPSI_Contnrs;                        //Delphi RTL Container of Classes
21231: 31 unit uPSI_MaskUtils;                      //RTL Edit and Mask functions
21232: 32 unit uPSI_MyBigInt;                       //big integer class with Math
21233: 33 unit uPSI_ConvUtils;                      //Delphi VCL Conversions engine
21234: 34 unit Types_Variants;                      //Delphi\Win32\rtl\sys
21235: 35 unit uPSI_IdHashSHA1;                     //Indy Crypto Lib
21236: 36 unit uPSI_IdHashMessageDigest            //Indy Crypto;
21237: 37 unit uPSI_IdASN1Util;                     //Indy ASN1Utility Routines;
21238: 38 unit uPSI_IdLogFile;                      //Indy Logger from LogBase
21239: 39 unit uPSI_IdIcmpClient;                   //Indy Ping ICMP
21240: 40 unit uPSI_IdHashMessageDigest_max        //Indy Crypto &OpenSSL;
21241: 41 unit uPSI_FileCtrl;                       //Delphi RTL
21242: 42 unit uPSI_Outline;                       //Delphi VCL
21243: 43 unit uPSI_ScktComp;                      //Delphi RTL
21244: 44 unit uPSI_Calendar;                      //Delphi VCL
21245: 45 unit uPSI_VListView                      //VListView;
21246: 46 unit uPSI_DBGrids;                       //Delphi VCL
21247: 47 unit uPSI_DBCtrls;                       //Delphi VCL
21248: 48 unit ide_debugoutput;                    //maXbox
21249: 49 unit uPSI_ComCtrls;                      //Delphi VCL
21250: 50 unit uPSC_stdctrls+;                     //Delphi VCL
21251: 51 unit uPSI_Dialogs;                       //Delphi VCL
21252: 52 unit uPSI_StdConvs;                      //Delphi RTL
21253: 53 unit uPSI_DBClient;                      //Delphi RTL
21254: 54 unit uPSI_DBPlatform;                    //Delphi RTL
21255: 55 unit uPSI_Provider;                      //Delphi RTL
21256: 56 unit uPSI_FMTBcd;                        //Delphi RTL
21257: 57 unit uPSI_DBCGrids;                      //Delphi VCL
21258: 58 unit uPSI_CDSUtil;                       //MIDAS
21259: 59 unit uPSI_VarHlpr;                       //Delphi RTL
21260: 60 unit uPSI_ExtDlgs;                       //Delphi VCL
21261: 61 unit sdpStopwatch;                       //maXbox
21262: 62 unit uPSI_JclStatistics;                 //JCL
21263: 63 unit uPSI_JclLogic;                      //JCL
21264: 64 unit uPSI_JclMiscel;                     //JCL
21265: 65 unit uPSI_JclMath_max;                   //JCL RTL
21266: 66 unit uPSI_uTPLb_StreamUtils;             //LockBox 3
21267: 67 unit uPSI_MathUtils;                     //BCB
21268: 68 unit uPSI_JclMultimedia;                 //JCL
21269: 69 unit uPSI_WideStrUtils;                  //Delphi API/RTL
21270: 70 unit uPSI_GraphUtil;                     //Delphi RTL
21271: 71 unit uPSI_TypeTrans;                     //Delphi RTL
21272: 72 unit uPSI_HTTPApp;                       //Delphi VCL
21273: 73 unit uPSI_DBWeb;                         //Delphi VCL
21274: 74 unit uPSI_DBBdeWeb;                      //Delphi VCL
21275: 75 unit uPSI_DBXpressWeb;                   //Delphi VCL
21276: 76 unit uPSI_ShadowWnd;                     //Delphi VCL
21277: 77 unit uPSI_ToolWin;                       //Delphi VCL
21278: 78 unit uPSI_Tabs;                          //Delphi VCL
21279: 79 unit uPSI_JclGraphUtils;                 //JCL
21280: 80 unit uPSI_JclCounter;                    //JCL
21281: 81 unit uPSI_JclSysInfo;                    //JCL
21282: 82 unit uPSI_JclSecurity;                   //JCL
21283: 83 unit uPSI_JclFileUtils;                  //JCL
21284: 84 unit uPSI_IdUserAccounts;                //Indy
21285: 85 unit uPSI_IdAuthentication;              //Indy
21286: 86 unit uPSI_uTPLb_AES;                     //LockBox 3
21287: 87 unit uPSI_IdHashSHA1;                    //LockBox 3
21288: 88 unit uTPLb_BlockCipher;                  //LockBox 3
21289: 89 unit uPSI_ValEdit.pas;                   //Delphi VCL
21290: 90 unit uPSI_JvVCLUtils;                    //JCL
21291: 91 unit uPSI_JvDBUtil;                      //JCL
21292: 92 unit uPSI_JvDBUtils;                     //JCL
21293: 93 unit uPSI_JvAppUtils;                    //JCL
21294: 94 unit uPSI_JvCtrlUtils;                   //JCL
21295: 95 unit uPSI_JvFormToHtml;                  //JCL
21296: 96 unit uPSI_JvParsing;                     //JCL
21297: 97 unit uPSI_SerDlgs;                       //Toolbox
21298: 98 unit uPSI_Serial;                        //Toolbox
21299: 99 unit uPSI_JvComponent;                   //JCL
21300: 100 unit uPSI_JvCalc;                       //JCL
21301: 101 unit uPSI_JvBdeUtils;                   //JCL
21302: 102 unit uPSI_JvDateUtil;                   //JCL
21303: 103 unit uPSI_JvGenetic;                    //JCL
21304: 104 unit uPSI_JclBase;                      //JCL
21305: 105 unit uPSI_JvUtils;                      //JCL
21306: 106 unit uPSI_JvStrUtil;                    //JCL
21307: 107 unit uPSI_JvStrUtils;                   //JCL
21308: 108 unit uPSI_JvFileUtil;                   //JCL
21309: 109 unit uPSI_JvMemoryInfos;                //JCL
21310: 110 unit uPSI_JvComputerInfo;               //JCL
21311: 111 unit uPSI_JvgCommClasses;               //JCL
21312: 112 unit uPSI_JvgLogics;                    //JCL
21313: 113 unit uPSI_JvLED;                        //JCL
21314: 114 unit uPSI_JvTurtle;                     //JCL
21315: 115 unit uPSI_SortThds; unit uPSI_ThSort;   //maXbox
21316: 116 unit uPSI_JvgUtils;                     //JCL
21317: 117 unit uPSI_JvExprParser;                 //JCL
21318: 118 unit uPSI_HexDump;                      //Borland
```

```
21319: 119 unit uPSI_DBLogDlg;                    //VCL
21320: 120 unit uPSI_SqlTimSt;                    //RTL
21321: 121 unit uPSI_JvHtmlParser;                //JCL
21322: 122 unit uPSI_JvgXMLSerializer;           //JCL
21323: 123 unit uPSI_JvJCLUtils;                 //JCL
21324: 124 unit uPSI_JvStrings;                   //JCL
21325: 125 unit uPSI_uTPLb_IntegerUtils;          //TurboPower
21326: 126 unit uPSI_uTPLb_HugeCardinal;          //TurboPower
21327: 127 unit uPSI_uTPLb_HugeCardinalUtils;     //TurboPower
21328: 128 unit uPSI_SynRegExpr;                  //SynEdit
21329: 129 unit uPSI_StUtils;                     //SysTools4
21330: 130 unit uPSI_StToHTML;                    //SysTools4
21331: 131 unit uPSI_StStrms;                     //SysTools4
21332: 132 unit uPSI_StFIN;                       //SysTools4
21333: 133 unit uPSI_StAstroP;                    //SysTools4
21334: 134 unit uPSI_StStat;                      //SysTools4
21335: 135 unit uPSI_StNetCon;                    //SysTools4
21336: 136 unit uPSI_StDecMth;                    //SysTools4
21337: 137 unit uPSI_StOStr;                      //SysTools4
21338: 138 unit uPSI_StPtrns;                     //SysTools4
21339: 139 unit uPSI_StNetMsg;                    //SysTools4
21340: 140 unit uPSI_StMath;                      //SysTools4
21341: 141 unit uPSI_StExpEng;                    //SysTools4
21342: 142 unit uPSI_StCRC;                       //SysTools4
21343: 143 unit uPSI_StExport,                    //SysTools4
21344: 144 unit uPSI_StExpLog,                    //SysTools4
21345: 145 unit uPSI_ActnList;                    //Delphi VCL
21346: 146 unit uPSI_jpeg;                        //Borland
21347: 147 unit uPSI_StRandom;                    //SysTools4
21348: 148 unit uPSI_StDict;                      //SysTools4
21349: 149 unit uPSI_StBCD;                       //SysTools4
21350: 150 unit uPSI_StTxtDat;                    //SysTools4
21351: 151 unit uPSI_StRegEx;                     //SysTools4
21352: 152 unit uPSI_IMouse;                      //VCL
21353: 153 unit uPSI_SyncObjs;                    //VCL
21354: 154 unit uPSI_AsyncCalls;                  //Hausladen
21355: 155 unit uPSI_ParallelJobs;                //Saraiva
21356: 156 unit uPSI_Variants;                    //VCL
21357: 157 unit uPSI_VarCmplx;                    //VCL Wolfram
21358: 158 unit uPSI_DTDSchema;                   //VCL
21359: 159 unit uPSI_ShLwApi;                     //Brakel
21360: 160 unit uPSI_IBUtils;                     //VCL
21361: 161 unit uPSI_CheckLst;                    //VCL
21362: 162 unit uPSI_JvSimpleXml;                 //JCL
21363: 163 unit uPSI_JclSimpleXml;                //JCL
21364: 164 unit uPSI_JvXmlDatabase;               //JCL
21365: 165 unit uPSI_JvMaxPixel;                  //JCL
21366: 166 unit uPSI_JvItemsSearchs;              //JCL
21367: 167 unit uPSI_StExpEng2;                   //SysTools4
21368: 168 unit uPSI_StGenLog;                    //SysTools4
21369: 169 unit uPSI_JvLogFile;                   //Jcl
21370: 170 unit uPSI_CPort;                       //ComPort Lib v4.11
21371: 171 unit uPSI_CPortCtl;                    //ComPort
21372: 172 unit uPSI_CPortEsc;                    //ComPort
21373: 173 unit BarCodeScaner;                    //ComPort
21374: 174 unit uPSI_JvGraph;                     //JCL
21375: 175 unit uPSI_JvComCtrls;                  //JCL
21376: 176 unit uPSI_GUITesting;                  //D Unit
21377: 177 unit uPSI_JvFindFiles;                 //JCL
21378: 178 unit uPSI_StSystem;                    //SysTools4
21379: 179 unit uPSI_JvKeyboardStates;            //JCL
21380: 180 unit uPSI_JvMail;                      //JCL
21381: 181 unit uPSI_JclConsole;                  //JCL
21382: 182 unit uPSI_JclLANMan;                   //JCL
21383: 183 unit uPSI_IdCustomHTTPServer;          //Indy
21384: 184 unit IdHTTPServer                      //Indy
21385: 185 unit uPSI_IdTCPServer;                 //Indy
21386: 186 unit uPSI_IdSocketHandle;              //Indy
21387: 187 unit uPSI_IdIOHandlerSocket;           //Indy
21388: 188 unit IdIOHandler;                      //Indy
21389: 189 unit uPSI_cutils;                      //Bloodshed
21390: 190 unit uPSI_BoldUtils;                   //boldsoft
21391: 191 unit uPSI_IdSimpleServer;              //Indy
21392: 192 unit uPSI_IdSSLOpenSSL;                //Indy
21393: 193 unit uPSI_IdMultipartFormData;         //Indy
21394: 194 unit uPSI_SynURIOpener;                //SynEdit
21395: 195 unit uPSI_PerlRegEx;                   //PCRE
21396: 196 unit uPSI_IdHeaderList;                //Indy
21397: 197 unit uPSI_StFirst;                     //SysTools4
21398: 198 unit uPSI_JvCtrls;                     //JCL
21399: 199 unit uPSI_IdTrivialFTPBase;            //Indy
21400: 200 unit uPSI_IdTrivialFTP;                //Indy
21401: 201 unit uPSI_IdUDPBase;                   //Indy
21402: 202 unit uPSI_IdUDPClient;                 //Indy
21403: 203 unit uPSI_utypes;                      //for DMath.DLL
21404: 204 unit uPSI_ShellAPI;                    //Borland
21405: 205 unit uPSI_IdRemoteCMDClient;           //Indy
21406: 206 unit uPSI_IdRemoteCMDServer;           //Indy
21407: 207 unit IdRexecServer;                    //Indy
```

```
21408: 208 unit IdRexec; (unit uPSI_IdRexec;)          //Indy
21409: 209 unit IdUDPServer;                          //Indy
21410: 210 unit IdTimeUDPServer;                      //Indy
21411: 211 unit IdTimeServer;                         //Indy
21412: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)   //Indy
21413: 213 unit uPSI_IdIPWatch;                       //Indy
21414: 214 unit uPSI_IdIrcServer;                     //Indy
21415: 215 unit uPSI_IdMessageCollection;             //Indy
21416: 216 unit uPSI_cPEM;                            //Fundamentals 4
21417: 217 unit uPSI_cFundamentUtils;                 //Fundamentals 4
21418: 218 unit uPSI_uwinplot;                        //DMath
21419: 219 unit uPSI_xrtl_util_CPUUtils;              //ExtentedRTL
21420: 220 unit uPSI_GR32_System;                     //Graphics32
21421: 221 unit uPSI_cFileUtils;                      //Fundamentals 4
21422: 222 unit uPSI_cDateTime; (timemachine)         //Fundamentals 4
21423: 223 unit uPSI_cTimers; (high precision timer)  //Fundamentals 4
21424: 224 unit uPSI_cRandom;                         //Fundamentals 4
21425: 225 unit uPSI_ueval;                           //DMath
21426: 226 unit uPSI_xrtl_net_URIUtils;               //ExtendedRTL
21427: 227 unit xrtl_net_URIUtils;                    //ExtendedRTL
21428: 228 unit uPSI_ufft;  (FFT)                     //DMath
21429: 229 unit uPSI_DBXChannel;                      //Delphi
21430: 230 unit uPSI_DBXIndyChannel;                  //Delphi Indy
21431: 231 unit uPSI_xrtl_util_COMCat;                //ExtendedRTL
21432: 232 unit uPSI_xrtl_util_StrUtils;              //ExtendedRTL
21433: 233 unit uPSI_xrtl_util_VariantUtils;          //ExtendedRTL
21434: 234 unit uPSI_xrtl_util_FileUtils;             //ExtendedRTL
21435: 235 unit xrtl_util_Compat;                     //ExtendedRTL
21436: 236 unit uPSI_OleAuto;                         //Borland
21437: 237 unit uPSI_xrtl_util_COMUtils;              //ExtendedRTL
21438: 238 unit uPSI_CmAdmCtl;                        //Borland
21439: 239 unit uPSI_ValEdit2;                        //VCL
21440: 240 unit uPSI_GR32;  //Graphics32             //Graphics32
21441: 241 unit uPSI_GR32_Image;                      //Graphics32
21442: 242 unit uPSI_xrtl_util_TimeUtils;             //ExtendedRTL
21443: 243 unit uPSI_xrtl_util_TimeZone;              //ExtendedRTL
21444: 244 unit uPSI_xrtl_util_TimeStamp;             //ExtendedRTL
21445: 245 unit uPSI_xrtl_util_Map;                   //ExtendedRTL
21446: 246 unit uPSI_xrtl_util_Set;                   //ExtendedRTL
21447: 247 unit uPSI_CPortMonitor;                    //ComPort
21448: 248 unit uPSI_StIniStm;                        //SysTools4
21449: 249 unit uPSI_GR32_ExtImage;                   //Graphics32
21450: 250 unit uPSI_GR32_OrdinalMaps;                //Graphics32
21451: 251 unit uPSI_GR32_Rasterizers;                //Graphics32
21452: 252 unit uPSI_xrtl_util_Exception;             //ExtendedRTL
21453: 253 unit uPSI_xrtl_util_Value;                 //ExtendedRTL
21454: 254 unit uPSI_xrtl_util_Compare;               //ExtendedRTL
21455: 255 unit uPSI_FlatSB;                          //VCL
21456: 256 unit uPSI_JvAnalogClock;                   //JCL
21457: 257 unit uPSI_JvAlarms;                        //JCL
21458: 258 unit uPSI_JvSQLS;                          //JCL
21459: 259 unit uPSI_JvDBSecur;                       //JCL
21460: 260 unit uPSI_JvDBQBE;                         //JCL
21461: 261 unit uPSI_JvStarfield;                     //JCL
21462: 262 unit uPSI_JVCLMiscal;                      //JCL
21463: 263 unit uPSI_JvProfiler32;                    //JCL
21464: 264 unit uPSI_JvDirectories,                   //JCL
21465: 265 unit uPSI_JclSchedule,                     //JCL
21466: 266 unit uPSI_JclSvcCtrl,                      //JCL
21467: 267 unit uPSI_JvSoundControl,                  //JCL
21468: 268 unit uPSI_JvBDESQLScript,                  //JCL
21469: 269 unit uPSI_JvgDigits,                       //JCL>
21470: 270 unit uPSI_ImgList;                         //TCustomImageList
21471: 271 unit uPSI_JclMIDI;                         //JCL>
21472: 272 unit uPSI_JclWinMidi;                      //JCL>
21473: 273 unit uPSI_JclNTFS;                         //JCL>
21474: 274 unit uPSI_JclAppInst;                      //JCL>
21475: 275 unit uPSI_JvRle;                           //JCL>
21476: 276 unit uPSI_JvRas32;                         //JCL>
21477: 277 unit uPSI_JvImageDrawThread,               //JCL>
21478: 278 unit uPSI_JvImageWindow,                   //JCL>
21479: 279 unit uPSI_JvTransparentForm;               //JCL>
21480: 280 unit uPSI_JvWinDialogs;                    //JCL>
21481: 281 unit uPSI_JvSimLogic,                      //JCL>
21482: 282 unit uPSI_JvSimIndicator,                  //JCL>
21483: 283 unit uPSI_JvSimPID,                        //JCL>
21484: 284 unit uPSI_JvSimPIDLinker,                  //JCL>
21485: 285 unit uPSI_IdRFCReply;                      //Indy
21486: 286 unit uPSI_IdIdent;                         //Indy
21487: 287 unit uPSI_IdIdentServer;                   //Indy
21488: 288 unit uPSI_JvPatchFile;                     //JCL
21489: 289 unit uPSI_StNetPfm;                        //SysTools4
21490: 290 unit uPSI_StNet;                           //SysTools4
21491: 291 unit uPSI_JclPeImage;                      //JCL
21492: 292 unit uPSI_JclPrint;                        //JCL
21493: 293 unit uPSI_JclMime;                         //JCL
21494: 294 unit uPSI_JvRichEdit;                      //JCL
21495: 295 unit uPSI_JvDBRichEd;                      //JCL
21496: 296 unit uPSI_JvDice;                          //JCL
```

```
21497: 297 unit uPSI_JvFloatEdit;                    //JCL 3.9.8
21498: 298 unit uPSI_JvDirFrm;                       //JCL
21499: 299 unit uPSI_JvDualList;                     //JCL
21500: 300 unit uPSI_JvSwitch;                       ////JCL
21501: 301 unit uPSI_JvTimerLst;                     ////JCL
21502: 302 unit uPSI_JvMemTable;                     //JCL
21503: 303 unit uPSI_JvObjStr;                       //JCL
21504: 304 unit uPSI_StLArr;                         //SysTools4
21505: 305 unit uPSI_StWmDCpy;                       //SysTools4
21506: 306 unit uPSI_StText;                         //SysTools4
21507: 307 unit uPSI_StNTLog;                        //SysTools4
21508: 308 unit uPSI_xrtl_math_Integer;              //ExtendedRTL
21509: 309 unit uPSI_JvImagPrvw;                     //JCL
21510: 310 unit uPSI_JvFormPatch;                    //JCL
21511: 311 unit uPSI_JvPicClip;                      //JCL
21512: 312 unit uPSI_JvDataConv;                     //JCL
21513: 313 unit uPSI_JvCpuUsage;                     //JCL
21514: 314 unit uPSI_JclUnitConv_mX2;                //JCL
21515: 315 unit JvDualListForm;                      //JCL
21516: 316 unit uPSI_JvCpuUsage2;                    //JCL
21517: 317 unit uPSI_JvParserForm;                   //JCL
21518: 318 unit uPSI_JvJanTreeView;                  //JCL
21519: 319 unit uPSI_JvTransLED;                     //JCL
21520: 320 unit uPSI_JvPlaylist;                     //JCL
21521: 321 unit uPSI_JvFormAutoSize;                 //JCL
21522: 322 unit uPSI_JvYearGridEditForm;             //JCL
21523: 323 unit uPSI_JvMarkupCommon;                 //JCL
21524: 324 unit uPSI_JvChart;                        //JCL
21525: 325 unit uPSI_JvXPCore;                       //JCL
21526: 326 unit uPSI_JvXPCoreUtils;                  //JCL
21527: 327 unit uPSI_StatsClasses;                   //mX4
21528: 328 unit uPSI_ExtCtrls2;                      //VCL
21529: 329 unit uPSI_JvUrlGrabbers;                  //JCL
21530: 330 unit uPSI_JvXmlTree;                      //JCL
21531: 331 unit uPSI_JvWavePlayer;                   //JCL
21532: 332 unit uPSI_JvUnicodeCanvas;                //JCL
21533: 333 unit uPSI_JvTFUtils;                      //JCL
21534: 334 unit uPSI_IdServerIOHandler;              //Indy
21535: 335 unit uPSI_IdServerIOHandlerSocket;        //Indy
21536: 336 unit uPSI_IdMessageCoder;                 //Indy
21537: 337 unit uPSI_IdMessageCoderMIME;             //Indy
21538: 338 unit uPSI_IdMIMETypes;                    //Indy
21539: 339 unit uPSI_JvConverter;                    //JCL
21540: 340 unit uPSI_JvCsvParse;                     //JCL
21541: 341 unit uPSI_umath;  unit uPSI_ugamma;       //DMath
21542: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
21543: 343 unit uPSI_JvDBGridExport;                 //JCL
21544: 344 unit uPSI_JvgExport;                      //JCL
21545: 345 unit uPSI_JvSerialMaker;                  //JCL
21546: 346 unit uPSI_JvWin32;                        //JCL
21547: 347 unit uPSI_JvPaintFX;                      //JCL
21548: 348 unit uPSI_JvOracleDataSet; (beta)         //JCL
21549: 349 unit uPSI_JvValidators; (preview)         //JCL
21550: 350 unit uPSI_JvNTEventLog;                   //JCL
21551: 351 unit uPSI_ShellZipTool;                   //mX4
21552: 352 unit uPSI_JvJoystick;                     //JCL
21553: 353 unit uPSI_JvMailSlots;                    //JCL
21554: 354 unit uPSI_JclComplex;                     //JCL
21555: 355 unit uPSI_SynPdf;                         //Synopse
21556: 356 unit uPSI_Registry;                       //VCL
21557: 357 unit uPSI_TlHelp32;                       //VCL
21558: 358 unit uPSI_JclRegistry;                    //JCL
21559: 359 unit uPSI_JvAirBrush;                     //JCL
21560: 360 unit uPSI_mORMotReport;                   //Synopse
21561: 361 unit uPSI_JclLocales;                     //JCL
21562: 362 unit uPSI_SynEdit;                        //SynEdit
21563: 363 unit uPSI_SynEditTypes;                   //SynEdit
21564: 364 unit uPSI_SynMacroRecorder;               //SynEdit
21565: 365 unit uPSI_LongIntList;                    //SynEdit
21566: 366 unit uPSI_devcutils;                      //DevC
21567: 367 unit uPSI_SynEditMiscClasses;             //SynEdit
21568: 368 unit uPSI_SynEditRegexSearch;             //SynEdit
21569: 369 unit uPSI_SynEditHighlighter;             //SynEdit
21570: 370 unit uPSI_SynHighlighterPas;              //SynEdit
21571: 371 unit uPSI_JvSearchFiles;                  //JCL
21572: 372 unit uPSI_SynHighlighterAny;              //Lazarus
21573: 373 unit uPSI_SynEditKeyCmds;                 //SynEdit
21574: 374 unit uPSI_SynEditMiscProcs,               //SynEdit
21575: 375 unit uPSI_SynEditKbdHandler               //SynEdit
21576: 376 unit uPSI_JvAppInst,                      //JCL
21577: 377 unit uPSI_JvAppEvent;                     //JCL
21578: 378 unit uPSI_JvAppCommand;                   //JCL
21579: 379 unit uPSI_JvAnimTitle;                    //JCL
21580: 380 unit uPSI_JvAnimatedImage;                //JCL
21581: 381 unit uPSI_SynEditExport;                  //SynEdit
21582: 382 unit uPSI_SynExportHTML;                  //SynEdit
21583: 383 unit uPSI_SynExportRTF;                   //SynEdit
21584: 384 unit uPSI_SynEditSearch;                  //SynEdit
21585: 385 unit uPSI_fMain_back                      //maXbox;
```

```
21586: 386 unit uPSI_JvZoom;                          //JCL
21587: 387 unit uPSI_PMrand;                          //PM
21588: 388 unit uPSI_JvSticker;                       //JCL
21589: 389 unit uPSI_XmlVerySimple;                   //mX4
21590: 390 unit uPSI_Services;                        //ExtPascal
21591: 391 unit uPSI_ExtPascalUtils;                  //ExtPascal
21592: 392 unit uPSI_SocketsDelphi;                   //ExtPascal
21593: 393 unit uPSI_StBarC;                         //SysTools
21594: 394 unit uPSI_StDbBarC;                       //SysTools
21595: 395 unit uPSI_StBarPN;                        //SysTools
21596: 396 unit uPSI_StDbPNBC;                       //SysTools
21597: 397 unit uPSI_StDb2DBC;                       //SysTools
21598: 398 unit uPSI_StMoney;                        //SysTools
21599: 399 unit uPSI_JvForth;                        //JCL
21600: 400 unit uPSI_RestRequest;                    //mX4
21601: 401 unit uPSI_HttpRESTConnectionIndy;         //mX4
21602: 402 unit uPSI_JvXmlDatabase; //update        //JCL
21603: 403 unit uPSI_StAstro;                        //SysTools
21604: 404 unit uPSI_StSort;                         //SysTools
21605: 405 unit uPSI_StDate;                         //SysTools
21606: 406 unit uPSI_StDateSt;                       //SysTools
21607: 407 unit uPSI_StBase;                         //SysTools
21608: 408 unit uPSI_StVInfo;                        //SysTools
21609: 409 unit uPSI_JvBrowseFolder;                 //JCL
21610: 410 unit uPSI_JvBoxProcs;                     //JCL
21611: 411 unit uPSI_urandom; (unit uranuvag;)       //DMath
21612: 412 unit uPSI_usimann; (unit ugenalg;)        //DMath
21613: 413 unit uPSI_JvHighlighter;                  //JCL
21614: 414 unit uPSI_Diff;                           //mX4
21615: 415 unit uPSI_SpringWinAPI;                   //DSpring
21616: 416 unit uPSI_StBits;                         //SysTools
21617: 417 unit uPSI_TomDBQue;                       //mX4
21618: 418 unit uPSI_MultilangTranslator;            //mX4
21619: 419 unit uPSI_HyperLabel;                     //mX4
21620: 420 unit uPSI_Starter;                        //mX4
21621: 421 unit uPSI_FileAssocs;                     //devC
21622: 422 unit uPSI_devFileMonitorX;                //devC
21623: 423 unit uPSI_devrun;                         //devC
21624: 424 unit uPSI_devExec;                        //devC
21625: 425 unit uPSI_oysUtils;                       //devC
21626: 426 unit uPSI_DosCommand;                     //devC
21627: 427 unit uPSI_CppTokenizer;                   //devC
21628: 428 unit uPSI_JvHLParser;                     //devC
21629: 429 unit uPSI_JclMapi;                        //JCL
21630: 430 unit uPSI_JclShell;                       //JCL
21631: 431 unit uPSI_JclCOM;                         //JCL
21632: 432 unit uPSI_GR32_Math;                      //Graphics32
21633: 433 unit uPSI_GR32_LowLevel;                  //Graphics32
21634: 434 unit uPSI_SimpleHl;                       //mX4
21635: 435 unit uPSI_GR32_Filters,                   //Graphics32
21636: 436 unit uPSI_GR32_VectorMaps;                //Graphics32
21637: 437 unit uPSI_cXMLFunctions;                  //Fundamentals 4
21638: 438 unit uPSI_JvTimer;                       //JCL
21639: 439 unit uPSI_cHTTPUtils;                    //Fundamentals 4
21640: 440 unit uPSI_cTLSUtils;                     //Fundamentals 4
21641: 441 unit uPSI_JclGraphics;                   //JCL
21642: 442 unit uPSI_JclSynch;                      //JCL
21643: 443 unit uPSI_IdTelnet;                      //Indy
21644: 444 unit uPSI_IdTelnetServer,                //Indy
21645: 445 unit uPSI_IdEcho,                        //Indy
21646: 446 unit uPSI_IdEchoServer,                  //Indy
21647: 447 unit uPSI_IdEchoUDP,                     //Indy
21648: 448 unit uPSI_IdEchoUDPServer,               //Indy
21649: 449 unit uPSI_IdSocks,                       //Indy
21650: 450 unit uPSI_IdAntiFreezeBase;              //Indy
21651: 451 unit uPSI_IdHostnameServer;              //Indy
21652: 452 unit uPSI_IdTunnelCommon,                //Indy
21653: 453 unit uPSI_IdTunnelMaster,                //Indy
21654: 454 unit uPSI_IdTunnelSlave,                 //Indy
21655: 455 unit uPSI_IdRSH,                         //Indy
21656: 456 unit uPSI_IdRSHServer,                   //Indy
21657: 457 unit uPSI_Spring_Cryptography_Utils;     //Spring4Delphi
21658: 458 unit uPSI_MapReader,                     //devC
21659: 459 unit uPSI_LibTar,                        //devC
21660: 460 unit uPSI_IdStack;                       //Indy
21661: 461 unit uPSI_IdBlockCipherIntercept;        //Indy
21662: 462 unit uPSI_IdChargenServer;               //Indy
21663: 463 unit uPSI_IdFTPServer,                   //Indy
21664: 464 unit uPSI_IdException,                   //Indy
21665: 465 unit uPSI_utexplot;                      //DMath
21666: 466 unit uPSI_uwinstr;                       //DMath
21667: 467 unit uPSI_VarRecUtils;                   //devC
21668: 468 unit uPSI_JvStringListToHtml,            //JCL
21669: 469 unit uPSI_JvStringHolder,                //JCL
21670: 470 unit uPSI_IdCoder;                       //Indy
21671: 471 unit uPSI_SynHighlighterDfm;             //Synedit
21672: 472 unit uHighlighterProcs; in 471           //Synedit
21673: 473 unit uPSI_LazFileUtils,                  //LCL
21674: 474 unit uPSI_IDECmdLine;                    //LCL
```

```
21675: 475 unit uPSI_lazMasks;                    //LCL
21676: 476 unit uPSI_ip_misc;                     //mX4
21677: 477 unit uPSI_Barcode;                     //LCL
21678: 478 unit uPSI_SimpleXML;                   //LCL
21679: 479 unit uPSI_JclIniFiles;                 //JCL
21680: 480 unit uPSI_D2XXUnit;  {$X-}             //FTDI
21681: 481 unit uPSI_JclDateTime;                 //JCL
21682: 482 unit uPSI_JclEDI;                      //JCL
21683: 483 unit uPSI_JclMiscel2;                  //JCL
21684: 484 unit uPSI_JclValidation;               //JCL
21685: 485 unit uPSI_JclAnsiStrings; {-PString}   //JCL
21686: 486 unit uPSI_SynEditMiscProcs2;           //Synedit
21687: 487 unit uPSI_JclStreams;                  //JCL
21688: 488 unit uPSI_QRCode;                      //mX4
21689: 489 unit uPSI_BlockSocket;                 //ExtPascal
21690: 490 unit uPSI_Masks,Utils                  //VCL
21691: 491 unit uPSI_synautil;                    //Synapse!
21692: 492 unit uPSI_JclMath_Class;               //JCL RTL
21693: 493 unit ugamdist; //Gamma function       //DMath
21694: 494 unit uibeta, ucorrel; //IBeta         //DMath
21695: 495 unit uPSI_SRMgr;                       //mX4
21696: 496 unit uPSI_HotLog;                      //mX4
21697: 497 unit uPSI_DebugBox;                    //mX4
21698: 498 unit uPSI_ustrings;                    //DMath
21699: 499 unit uPSI_uregtest;                    //DMath
21700: 500 unit uPSI_usimplex;                    //DMath
21701: 501 unit uPSI_uhyper;                      //DMath
21702: 502 unit uPSI_IdHL7;                       //Indy
21703: 503 unit uPSI_IdIPMCastBase,               //Indy
21704: 504 unit uPSI_IdIPMCastServer;             //Indy
21705: 505 unit uPSI_IdIPMCastClient;             //Indy
21706: 506 unit uPSI_unlfit; //nlregression      //DMath
21707: 507 unit uPSI_IdRawHeaders;                //Indy
21708: 508 unit uPSI_IdRawClient;                 //Indy
21709: 509 unit uPSI_IdRawFunctions;              //Indy
21710: 510 unit uPSI_IdTCPStream;                 //Indy
21711: 511 unit uPSI_IdSNPP;                      //Indy
21712: 512 unit uPSI_St2DBarC;                    //SysTools
21713: 513 unit uPSI_ImageWin;  //FTL            //VCL
21714: 514 unit uPSI_CustomDrawTreeView; //FTL    //VCL
21715: 515 unit uPSI_GraphWin;  //FTL            //VCL
21716: 516 unit uPSI_actionMain;  //FTL           //VCL
21717: 517 unit uPSI_StSpawn;                     //SysTools
21718: 518 unit uPSI_CtlPanel;                    //VCL
21719: 519 unit uPSI_IdLPR;                       //Indy
21720: 520 unit uPSI_SockRequestInterpreter;      //Indy
21721: 521 unit uPSI_ulambert;                    //DMath
21722: 522 unit uPSI_ucholesk;                    //DMath
21723: 523 unit uPSI_SimpleDS;                    //VCL
21724: 524 unit uPSI_DBXSqlScanner;               //VCL
21725: 525 unit uPSI_DBXMetaDataUtil;             //VCL
21726: 526 unit uPSI_Chart;                       //TEE
21727: 527 unit uPSI_TeeProcs;                    //TEE
21728: 528 unit mXBDEUtils;                       //mX4
21729: 529 unit uPSI_MDIEdit;                     //VCL
21730: 530 unit uPSI_CopyPrsr;                    //VCL
21731: 531 unit uPSI_SockApp;                     //VCL
21732: 532 unit uPSI_AppEvnts;                    //VCL
21733: 533 unit uPSI_ExtActns;                    //VCL
21734: 534 unit uPSI_TeEngine;                    //TEE
21735: 535 unit uPSI_CoolMain; //browser         //VCL
21736: 536 unit uPSI_StCRC;                       //SysTools
21737: 537 unit uPSI_StDecMth2;                   //SysTools
21738: 538 unit uPSI_frmExportMain;               //Synedit
21739: 539 unit uPSI_SynDBEdit;                   //Synedit
21740: 540 unit uPSI_SynEditWildcardSearch;       //Synedit
21741: 541 unit uPSI_BoldComUtils;                //BOLD
21742: 542 unit uPSI_BoldIsoDateTime;             //BOLD
21743: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils  //BOLD
21744: 544 unit uPSI_BoldXMLRequests;             //BOLD
21745: 545 unit uPSI_BoldStringList;              //BOLD
21746: 546 unit uPSI_BoldFileHandler;             //BOLD
21747: 547 unit uPSI_BoldContainers;              //BOLD
21748: 548 unit uPSI_BoldQueryUserDlg;            //BOLD
21749: 549 unit uPSI_BoldWinINet;                 //BOLD
21750: 550 unit uPSI_BoldQueue;                   //BOLD
21751: 551 unit uPSI_JvPcx;                       //JCL
21752: 552 unit uPSI_IdWhois;                     //Indy
21753: 553 unit uPSI_IdWhoIsServer;               //Indy
21754: 554 unit uPSI_IdGopher;                    //Indy
21755: 555 unit uPSI_IdDateTimeStamp;             //Indy
21756: 556 unit uPSI_IdDayTimeServer;             //Indy
21757: 557 unit uPSI_IdDayTimeUDP;                //Indy
21758: 558 unit uPSI_IdDayTimeUDPServer;          //Indy
21759: 559 unit uPSI_IdDICTServer;                //Indy
21760: 560 unit uPSI_IdDiscardServer;             //Indy
21761: 561 unit uPSI_IdDiscardUDPServer;          //Indy
21762: 562 unit uPSI_IdMappedFTP;                 //Indy
21763: 563 unit uPSI_IdMappedPortTCP;             //Indy
```

```
21764: 564 unit uPSI_IdGopherServer;                //Indy
21765: 565 unit uPSI_IdQotdServer;                  //Indy
21766: 566 unit uPSI_JvRgbToHtml;                   //JCL
21767: 567 unit uPSI_JvRemLog,                      //JCL
21768: 568 unit uPSI_JvSysComp;                     //JCL
21769: 569 unit uPSI_JvTMTL;                       //JCL
21770: 570 unit uPSI_JvWinampAPI;                  //JCL
21771: 571 unit uPSI_MSysUtils;                    //mX4
21772: 572 unit uPSI_ESBMaths;                     //ESB
21773: 573 unit uPSI_ESBMaths2;                    //ESB
21774: 574 unit uPSI_uLkJSON;                      //Lk
21775: 575 unit uPSI_ZURL;  //Zeos                 //Zeos
21776: 576 unit uPSI_ZSysUtils;                    //Zeos
21777: 577 unit unaUtils internals                 //UNA
21778: 578 unit uPSI_ZMatchPattern;                //Zeos
21779: 579 unit uPSI_ZClasses;                     //Zeos
21780: 580 unit uPSI_ZCollections;                 //Zeos
21781: 581 unit uPSI_ZEncoding;                    //Zeos
21782: 582 unit uPSI_IdRawBase;                    //Indy
21783: 583 unit uPSI_IdNTLM;                       //Indy
21784: 584 unit uPSI_IdNNTP;                       //Indy
21785: 585 unit uPSI_usniffer; //PortScanForm      //mX4
21786: 586 unit uPSI_IdCoderMIME;                  //Indy
21787: 587 unit uPSI_IdCoderUUE;                   //Indy
21788: 588 unit uPSI_IdCoderXXE;                   //Indy
21789: 589 unit uPSI_IdCoder3to4;                  //Indy
21790: 590 unit uPSI_IdCookie;                     //Indy
21791: 591 unit uPSI_IdCookieManager;              //Indy
21792: 592 unit uPSI_WDosSocketUtils;              //WDos
21793: 593 unit uPSI_WDosPlcUtils;                 //WDos
21794: 594 unit uPSI_WDosPorts;                    //WDos
21795: 595 unit uPSI_WDosResolvers;                //WDos
21796: 596 unit uPSI_WDosTimers;                   //WDos
21797: 597 unit uPSI_WDosPlcs;                     //WDos
21798: 598 unit uPSI_WDosPneumatics;               //WDos
21799: 599 unit uPSI_IdFingerServer;               //Indy
21800: 600 unit uPSI_IdDNSResolver;                //Indy
21801: 601 unit uPSI_IdHTTPWebBrokerBridge;        //Indy
21802: 602 unit uPSI_IdIntercept;                  //Indy
21803: 603 unit uPSI_IdIPMCastBase;                //Indy
21804: 604 unit uPSI_IdLogBase;                    //Indy
21805: 605 unit uPSI_IdIOHandlerStream;            //Indy
21806: 606 unit uPSI_IdMappedPortUDP;              //Indy
21807: 607 unit uPSI_IdQOTDUDPServer;              //Indy
21808: 608 unit uPSI_IdQOTDUDP;                    //Indy
21809: 609 unit uPSI_IdSysLog;                     //Indy
21810: 610 unit uPSI_IdSysLogServer;               //Indy
21811: 611 unit uPSI_IdSysLogMessage;              //Indy
21812: 612 unit uPSI_IdTimeServer;                 //Indy
21813: 613 unit uPSI_IdTimeUDP;                    //Indy
21814: 614 unit uPSI_IdTimeUDPServer;              //Indy
21815: 615 unit uPSI_IdUserAccounts;               //Indy
21816: 616 unit uPSI_TextUtils;                    //mX4
21817: 617 unit uPSI_MandelbrotEngine;             //mX4
21818: 618 unit uPSI_delphi_arduino_Unit1;         //mX4
21819: 619 unit uPSI_DTDSchema2;                   //mX4
21820: 620 unit uPSI_fplotMain;                    //DMath
21821: 621 unit uPSI_FindFileIter;                 //mX4
21822: 622 unit uPSI_PppState;  (JclStrHashMap)    //PPP
21823: 623 unit uPSI_PppParser;                    //PPP
21824: 624 unit uPSI_PppLexer;                     //PPP
21825: 625 unit uPSI_PCharUtils;                   //PPP
21826: 626 unit uPSI_uJSON;                        //WU
21827: 627 unit uPSI_JclStrHashMap;                //JCL
21828: 628 unit uPSI_JclHookExcept;                //JCL
21829: 629 unit uPSI_EncdDecd;                     //VCL
21830: 630 unit uPSI_SockAppReg;                   //VCL
21831: 631 unit uPSI_PJFileHandle;                 //PJ
21832: 632 unit uPSI_PJEnvVars;                    //PJ
21833: 633 unit uPSI_PJPipe;                       //PJ
21834: 634 unit uPSI_PJPipeFilters;                //PJ
21835: 635 unit uPSI_PJConsoleApp;                 //PJ
21836: 636 unit uPSI_UConsoleAppEx;                //PJ
21837: 637 unit uPSI_DbxSocketChannelNative,       //VCL
21838: 638 unit uPSI_DbxDataGenerator,             //VCL
21839: 639 unit uPSI_DBXClient;                    //VCL
21840: 640 unit uPSI_IdLogEvent;                   //Indy
21841: 641 unit uPSI_Reversi;                      //mX4
21842: 642 unit uPSI_Geometry;                     //mX4
21843: 643 unit uPSI_IdSMTPServer;                 //Indy
21844: 644 unit uPSI_Textures;                     //mX4
21845: 645 unit uPSI_IBX;                          //VCL
21846: 646 unit uPSI_IWDBCommon;                   //VCL
21847: 647 unit uPSI_SortGrid;                     //mX4
21848: 648 unit uPSI_IB;                           //VCL
21849: 649 unit uPSI_IBScript;                     //VCL
21850: 650 unit uPSI_JvCSVBaseControls;            //JCL
21851: 651 unit uPSI_Jvg3DColors;                  //JCL
21852: 652 unit uPSI_JvHLEditor;  //beat           //JCL
```

```
21853: 653 unit uPSI_JvShellHook;                    //JCL
21854: 654 unit uPSI_DBCommon2                       //VCL
21855: 655 unit uPSI_JvSHFileOperation;              //JCL
21856: 656 unit uPSI_uFilexport;                     //mX4
21857: 657 unit uPSI_JvDialogs;                      //JCL
21858: 658 unit uPSI_JvDBTreeView;                   //JCL
21859: 659 unit uPSI_JvDBUltimGrid;                  //JCL
21860: 660 unit uPSI_JvDBQueryParamsForm;            //JCL
21861: 661 unit uPSI_JvExControls;                   //JCL
21862: 662 unit uPSI_JvBDEMemTable;                  //JCL
21863: 663 unit uPSI_JvCommStatus;                   //JCL
21864: 664 unit uPSI_JvMailSlots2;                   //JCL
21865: 665 unit uPSI_JvgWinMask;                     //JCL
21866: 666 unit uPSI_StEclpse;                       //SysTools
21867: 667 unit uPSI_StMime;                         //SysTools
21868: 668 unit uPSI_StList;                         //SysTools
21869: 669 unit uPSI_StMerge;                        //SysTools
21870: 670 unit uPSI_StStrS;                         //SysTools
21871: 671 unit uPSI_StTree,                         //SysTools
21872: 672 unit uPSI_StVArr;                         //SysTools
21873: 673 unit uPSI_StRegIni;                       //SysTools
21874: 674 unit uPSI_urkf;                           //DMath
21875: 675 unit uPSI_usvd;                           //DMath
21876: 676 unit uPSI_DepWalkUtils;                   //JCL
21877: 677 unit uPSI_OptionsFrm;                     //JCL
21878: 678 unit yuvconverts;                         //mX4
21879: 679 uPSI_JvPropAutoSave;                      //JCL
21880: 680 uPSI_AclAPI;                              //alcinoe
21881: 681 uPSI_AviCap;                              //alcinoe
21882: 682 uPSI_ALAVLBinaryTree;                     //alcinoe
21883: 683 uPSI_ALFcnMisc;                           //alcinoe
21884: 684 uPSI_ALStringList;                        //alcinoe
21885: 685 uPSI_ALQuickSortList;                     //alcinoe
21886: 686 uPSI_ALStaticText;                        //alcinoe
21887: 687 uPSI_ALJSONDoc;                           //alcinoe
21888: 688 uPSI_ALGSMComm;                           //alcinoe
21889: 689 uPSI_ALWindows;                           //alcinoe
21890: 690 uPSI_ALMultiPartFormDataParser;           //alcinoe
21891: 691 uPSI_ALHttpCommon;                        //alcinoe
21892: 692 uPSI_ALWebSpider,                         //alcinoe
21893: 693 uPSI_ALHttpClient;                        //alcinoe
21894: 694 uPSI_ALFcnHTML;                           //alcinoe
21895: 695 uPSI_ALFTPClient;                         //alcinoe
21896: 696 uPSI_ALInternetMessageCommon;             //alcinoe
21897: 697 uPSI_ALWininetHttpClient;                 //alcinoe
21898: 698 uPSI_ALWinInetFTPClient;                  //alcinoe
21899: 699 uPSI_ALWinHttpWrapper;                    //alcinoe
21900: 700 uPSI_ALWinHttpClient;                     //alcinoe
21901: 701 uPSI_ALFcnWinSock;                        //alcinoe
21902: 702 uPSI_ALFcnSQL;                            //alcinoe
21903: 703 uPSI_ALFcnCGI;                            //alcinoe
21904: 704 uPSI_ALFcnExecute;                        //alcinoe
21905: 705 uPSI_ALFcnFile;                           //alcinoe
21906: 706 uPSI_ALFcnMime;                           //alcinoe
21907: 707 uPSI_ALPhpRunner;                         //alcinoe
21908: 708 uPSI_ALGraphic;                           //alcinoe
21909: 709 uPSI_ALIniFiles;                          //alcinoe
21910: 710 uPSI_ALMemCachedClient;                   //alcinoe
21911: 711 unit uPSI_MyGrids;                        //mX4
21912: 712 uPSI_ALMultiPartMixedParser               //alcinoe
21913: 713 uPSI_ALSMTPClient                         //alcinoe
21914: 714 uPSI_ALNNTPClient;                        //alcinoe
21915: 715 uPSI_ALHintBalloon;                       //alcinoe
21916: 716 unit uPSI_ALXmlDoc;                       //alcinoe
21917: 717 unit uPSI_IPCThrd;                        //VCL
21918: 718 unit uPSI_MonForm;                        //VCL
21919: 719 unit uPSI_TeCanvas;                       //Orpheus
21920: 720 unit uPSI_Ovcmisc;                        //Orpheus
21921: 721 unit uPSI_ovcfiler;                       //Orpheus
21922: 722 unit uPSI_ovcstate;                       //Orpheus
21923: 723 unit uPSI_ovccoco;                        //Orpheus
21924: 724 unit uPSI_ovcrvexp;                       //Orpheus
21925: 725 unit uPSI_OvcFormatSettings;              //Orpheus
21926: 726 unit uPSI_OvcUtils;                       //Orpheus
21927: 727 unit uPSI_ovcstore;                       //Orpheus
21928: 728 unit uPSI_ovcstr;                         //Orpheus
21929: 729 unit uPSI_ovcmru;                         //Orpheus
21930: 730 unit uPSI_ovccmd;                         //Orpheus
21931: 731 unit uPSI_ovctimer;                       //Orpheus
21932: 732 unit uPSI_ovcintl;                        //Orpheus
21933: 733 uPSI_AfCircularBuffer;                    //AsyncFree
21934: 734 uPSI_AfUtils;                             //AsyncFree
21935: 735 uPSI_AfSafeSync;                          //AsyncFree
21936: 736 uPSI_AfComPortCore;                       //AsyncFree
21937: 737 uPSI_AfComPort;                           //AsyncFree
21938: 738 uPSI_AfPortControls;                      //AsyncFree
21939: 739 uPSI_AfDataDispatcher;                    //AsyncFree
21940: 740 uPSI_AfViewers;                           //AsyncFree
21941: 741 uPSI_AfDataTerminal;                      //AsyncFree
```

```
21942: 742 uPSI_SimplePortMain;                    //AsyncFree
21943: 743 unit uPSI_ovcclock;                     //Orpheus
21944: 744 unit uPSI_o32intlst;                    //Orpheus
21945: 745 unit uPSI_o32ledlabel;                  //Orpheus
21946: 746 unit uPSI_AlMySqlClient;                //alcinoe
21947: 747 unit uPSI_ALFBXClient;                  //alcinoe
21948: 748 unit uPSI_ALFcnSQL;                     //alcinoe
21949: 749 unit uPSI_AsyncTimer;                   //mX4
21950: 750 unit uPSI_ApplicationFileIO;            //mX4
21951: 751 unit uPSI_PsAPI;                        //VCLé
21952: 752 uPSI_ovcuser;                           //Orpheus
21953: 753 uPSI_ovcurl;                            //Orpheus
21954: 754 uPSI_ovcvlb;                            //Orpheus
21955: 755 uPSI_ovccolor;                          //Orpheus
21956: 756 uPSI_ALFBXLib,                          //alcinoe
21957: 757 uPSI_ovcmeter;                          //Orpheus
21958: 758 uPSI_ovcpeakm;                          //Orpheus
21959: 759 uPSI_O32BGSty;                          //Orpheus
21960: 760 uPSI_ovcBidi;                           //Orpheus
21961: 761 uPSI_ovctcary;                          //Orpheus
21962: 762 uPSI_DXPUtils;                          //mX4
21963: 763 uPSI_ALMultiPartBaseParser;             //alcinoe
21964: 764 uPSI_ALMultiPartAlternativeParser;      //alcinoe
21965: 765 uPSI_ALPOP3Client;                      //alcinoe
21966: 766 uPSI_SmallUtils;                        //mX4
21967: 767 uPSI_MakeApp;                           //mX4
21968: 768 uPSI_O32MouseMon;                       //Orpheus
21969: 769 uPSI_OvcCache;                          //Orpheus
21970: 770 uPSI_ovccalc;                           //Orpheus
21971: 771 uPSI_Joystick,                          //OpenGL
21972: 772 uPSI_ScreenSaver;                       //OpenGL
21973: 773 uPSI_XCollection,                       //OpenGL
21974: 774 uPSI_Polynomials,                       //OpenGL
21975: 775 uPSI_PersistentClasses, //9.86         //OpenGL
21976: 776 uPSI_VectorLists;                       //OpenGL
21977: 777 uPSI_XOpenGL,                          //OpenGL
21978: 778 uPSI_MeshUtils;                         //OpenGL
21979: 779 unit uPSI_JclSysUtils;                  //JCL
21980: 780 unit uPSI_JclBorlandTools;              //JCL
21981: 781 unit JclFileUtils_max;                  //JCL
21982: 782 uPSI_AfDataControls,                    //AsyncFree
21983: 783 uPSI_GLSilhouette;                      //OpenGL
21984: 784 uPSI_JclSysUtils_class;                 //JCL
21985: 785 uPSI_JclFileUtils_class;                //JCL
21986: 786 uPSI_FileUtil;                          //JCL
21987: 787 uPSI_changefind;                        //mX4
21988: 788 uPSI_cmdIntf;                           //mX4
21989: 789 uPSI_fservice;                          //mX4
21990: 790 uPSI_Keyboard;                          //OpenGL
21991: 791 uPSI_VRMLParser,                        //OpenGL
21992: 792 uPSI_GLFileVRML,                        //OpenGL
21993: 793 uPSI_Octree;                            //OpenGL
21994: 794 uPSI_GLPolyhedron,                      //OpenGL
21995: 795 uPSI_GLCrossPlatform;                   //OpenGL
21996: 796 uPSI_GLParticles;                       //OpenGL
21997: 797 uPSI_GLNavigator;                       //OpenGL
21998: 798 uPSI_GLStarRecord;                      //OpenGL
21999: 799 uPSI_GLTextureCombiners;                //OpenGL
22000: 800 uPSI_GLCanvas;                          //OpenGL
22001: 801 uPSI_GeometryBB;                        //OpenGL
22002: 802 uPSI_GeometryCoordinates;               //OpenGL
22003: 803 uPSI_VectorGeometry;                    //OpenGL
22004: 804 uPSI_BumpMapping;                       //OpenGL
22005: 805 uPSI_TGA;                               //OpenGL
22006: 806 uPSI_GLVectorFileObjects;               //OpenGL
22007: 807 uPSI_IMM;                               //VCL
22008: 808 uPSI_CategoryButtons;                   //VCL
22009: 809 uPSI_ButtonGroup;                       //VCL
22010: 810 uPSI_DbExcept;                          //VCL
22011: 811 uPSI_AxCtrls;                           //VCL
22012: 812 uPSI_GL_actorUnit1;                     //OpenGL
22013: 813 uPSI_StdVCL;                            //VCL
22014: 814 unit CurvesAndSurfaces;                 //OpenGL
22015: 815 uPSI_DataAwareMain;                     //AsyncFree
22016: 816 uPSI_TabNotBk;                          //VCL
22017: 817 uPSI_udwsfiler;                         //mX4
22018: 818 uPSI_synaip;                            //Synapse!
22019: 819 uPSI_synacode;                          //Synapse
22020: 820 uPSI_synachar;                          //Synapse
22021: 821 uPSI_synamisc;                          //Synapse
22022: 822 uPSI_synaser;                           //Synapse
22023: 823 uPSI_synaicnv;                          //Synapse
22024: 824 uPSI_tlntsend;                          //Synapse
22025: 825 uPSI_pingsend;                          //Synapse
22026: 826 uPSI_blcksock;                          //Synapse
22027: 827 uPSI_asn1util;                          //Synapse
22028: 828 uPSI_dnssend;                           //Synapse
22029: 829 uPSI_clamsend;                          //Synapse
22030: 830 uPSI_ldapsend;                          //Synapse
```

```
22031: 831 uPSI_mimemess;                          //Synapse
22032: 832 uPSI_slogsend;                          //Synapse
22033: 833 uPSI_mimepart;                          //Synapse
22034: 834 uPSI_mimeinln;                          //Synapse
22035: 835 uPSI_ftpsend,                           //Synapse
22036: 836 uPSI_ftptsend;                          //Synapse
22037: 837 uPSI_httpsend;                          //Synapse
22038: 838 uPSI_sntpsend;                          //Synapse
22039: 839 uPSI_smtpsend;                          //Synapse
22040: 840 uPSI_snmpsend;                          //Synapse
22041: 841 uPSI_imapsend;                          //Synapse
22042: 842 uPSI_pop3send;                          //Synapse
22043: 843 uPSI_nntpsend;                          //Synapse
22044: 844 uPSI_ssl_cryptlib;                      //Synapse
22045: 845 uPSI_ssl_openssl;                       //Synapse
22046: 846 uPSI_synhttp_daemon;                    //Synapse
22047: 847 uPSI_NetWork;                           //mX4
22048: 848 uPSI_PingThread;                        //Synapse
22049: 849 uPSI_JvThreadTimer;                     //JCL
22050: 850 unit uPSI_wwSystem;                     //InfoPower
22051: 851 unit uPSI_IdComponent;                  //Indy
22052: 852 unit uPSI_IdIOHandlerThrottle;          //Indy
22053: 853 unit uPSI_Themes;                       //VCL
22054: 854 unit uPSI_StdStyleActnCtrls;            //VCL
22055: 855 unit uPSI_UDDIHelper;                   //VCL
22056: 856 unit uPSI_IdIMAP4Server;                //Indy
22057: 857 uPSI_VariantSymbolTable,                //VCL //3.9.9.92
22058: 858 uPSI_udf_glob,                          //mX4
22059: 859 uPSI_TabGrid,                           //VCL
22060: 860 uPSI_JsDBTreeView,                      //mX4
22061: 861 uPSI_JsSendMail,                       //mX4
22062: 862 uPSI_dbTvRecordList,                    //mX4
22063: 863 uPSI_TreeVwEx,                         //mX4
22064: 864 uPSI_ECDataLink,                       //mX4
22065: 865 uPSI_dbTree,                           //mX4
22066: 866 uPSI_dbTreeCBox,                       //mX4
22067: 867 unit uPSI_Debug; //TfrmDebug          //mX4
22068: 868 uPSI_TimeFncs;                         //mX4
22069: 869 uPSI_FileIntf,                         //VCL
22070: 870 uPSI_SockTransport,                    //RTL
22071: 871 unit uPSI_WinInet;                     //RTL
22072: 872 unit uPSI_Wwstr;                       //mX4
22073: 873 uPSI_DBLookup,                         //VCL
22074: 874 uPSI_Hotspot,                          //mX4
22075: 875 uPSI_HList; //History List             //mX4
22076: 876 unit uPSI_DrTable;                     //VCL
22077: 877 uPSI_TConnect,                         //VCL
22078: 878 uPSI_DataBkr,                          //VCL
22079: 879 uPSI_HTTPIntr;                         //VCL
22080: 880 unit uPSI_Mathbox;                     //mX4
22081: 881 uPSI_cyIndy,                           //cY
22082: 882 uPSI_cySysUtils,                       //cY
22083: 883 uPSI_cyWinUtils,                       //cY
22084: 884 uPSI_cyStrUtils,                       //cY
22085: 885 uPSI_cyObjUtils,                       //cY
22086: 886 uPSI_cyDateUtils,                      //cY
22087: 887 uPSI_cyBDE,                            //cY
22088: 888 uPSI_cyClasses,                        //cY
22089: 889 uPSI_cyGraphics,  //3.9.9.94_2         //cY
22090: 890 unit uPSI_cyTypes;                     //cY
22091: 891 uPSI_JvDateTimePicker,                 //JCL
22092: 892 uPSI_JvCreateProcess,                  //JCL
22093: 893 uPSI_JvEasterEgg,                      //JCL
22094: 894 uPSI_WinSvc,  //3.9.9.94_3            //VCL
22095: 895 uPSI_SvcMgr                           //VCL
22096: 896 unit uPSI_JvPickDate;                  //JCL
22097: 897 unit uPSI_JvNotify;                    //JCL
22098: 898 uPSI_JvStrHlder                        //JCL
22099: 899 unit uPSI_JclNTFS2;                    //JCL
22100: 900 uPSI_Jcl8087 //math coprocessor       //JCL
22101: 901 uPSI_JvAddPrinter                      //JCL
22102: 902 uPSI_JvCabFile                         //JCL
22103: 903 uPSI_JvDataEmbedded;                   //JCL
22104: 904 unit uPSI_U_HexView;                   //mX4
22105: 905 uPSI_UWavein4,                         //mX4
22106: 906 uPSI_AMixer,                           //mX4
22107: 907 uPSI_JvaScrollText,                    //mX4
22108: 908 uPSI_JvArrow,                         //mX4
22109: 909 unit uPSI_UrlMon;                      //mX4
22110: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'  //mX4
22111: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;  //DFF
22112: 912 unit uPSI_DFFUtils;                     //DFF
22113: 913 unit uPSI_MathsLib;                    //DFF
22114: 914 uPSI_UIntList;                         //DFF
22115: 915 uPSI_UGetParens;                       //DFF
22116: 916 unit uPSI_UGeometry;                   //DFF
22117: 917 unit uPSI_UAstronomy;                  //DFF
22118: 918 unit uPSI_UCardComponentV2;            //DFF
22119: 919 unit uPSI_UTGraphSearch;               //DFF
```

```
22120: 920 unit uPSI_UParser10;                    //DFF
22121: 921 unit uPSI_cyIEUtils;                    //cY
22122: 922 unit uPSI_UcomboV2;                     //DFF
22123: 923 uPSI_cyBaseComm,                        //cY
22124: 924 uPSI_cyAppInstances,                    //cY
22125: 925 uPSI_cyAttract,                         //cY
22126: 926 uPSI_cyDERUtils                         //cY
22127: 927 unit uPSI_cyDocER;                      //cY
22128: 928 unit uPSI_ODBC;                         //mX
22129: 929 unit uPSI_AssocExec;                    //mX
22130: 930 uPSI_cyBaseCommRoomConnector,           //cY
22131: 931 uPSI_cyCommRoomConnector,               //cY
22132: 932 uPSI_cyCommunicate,                     //cY
22133: 933 uPSI_cyImage;                           //cY
22134: 934 uPSI_cyBaseContainer                    //cY
22135: 935 uPSI_cyModalContainer,                  //cY
22136: 936 uPSI_cyFlyingContainer;                 //cY
22137: 937 uPSI_RegStr,                           //VCL
22138: 938 uPSI_HtmlHelpViewer;                    //VCL
22139: 939 unit uPSI_cyIniForm                     //cY
22140: 940 unit uPSI_cyVirtualGrid;               //cY
22141: 941 uPSI_Profiler,                          //DA
22142: 942 uPSI_BackgroundWorker,                  //DA
22143: 943 uPSI_WavePlay,                         //DA
22144: 944 uPSI_WaveTimer,                        //DA
22145: 945 uPSI_WaveUtils;                        //DA
22146: 946 uPSI_NamedPipes,                       //TB
22147: 947 uPSI_NamedPipeServer,                  //TB
22148: 948 unit uPSI_process,                     //TB
22149: 949 unit uPSI_DPUtils;                     //TB
22150: 950 unit uPSI_CommonTools;                 //TB
22151: 951 uPSI_DataSendToWeb,                    //TB
22152: 952 uPSI_StarCalc,                         //TB
22153: 953 uPSI_D2_XPVistaHelperU                 //TB
22154: 954 unit uPSI_NetTools                     //TB
22155: 955 unit uPSI_Pipes                        //TB
22156: 956 uPSI_ProcessUnit,                      //mX
22157: 957 uPSI_adGSM,                            //mX
22158: 958 unit uPSI_BetterADODataSet;            //mX
22159: 959 unit uPSI_AdSelCom; //FTT             //mX
22160: 960 unit unit uPSI_dwsXPlatform;           //DWS
22161: 961 uPSI_AdSocket;                         //mX Turbo Power
22162: 962 uPSI_AdPacket;                         //mX
22163: 963 uPSI_AdPort;                           //mX
22164: 964 uPSI_PathFunc;                         //Inno
22165: 965 uPSI_CmnFunc;                          //Inno
22166: 966 uPSI_CmnFunc2; //Inno Setup            //Inno
22167: 967 unit uPSI_BitmapImage;                 //mX4
22168: 968 unit uPSI_ImageGrabber;                //mX4
22169: 969 uPSI_SecurityFunc,                     //Inno
22170: 970 uPSI_RedirFunc,                        //Inno
22171: 971 uPSI_FIFO, (MemoryStream)              //mX4
22172: 972 uPSI_Int64Em,                          //Inno
22173: 973 unit uPSI_InstFunc;                    //Inno
22174: 974 unit uPSI_LibFusion;                   //Inno
22175: 975 uPSI_SimpleExpression;                 //Inno
22176: 976 uPSI_unitResourceDetails,              //XN
22177: 977 uPSI_unitResFile,                      //XN
22178: 978 unit uPSI_simpleComport;               //mX4
22179: 979 unit uPSI_AfViewershelpers;            //Async
22180: 980 unit uPSI_Console;                     //mX4
22181: 981 unit uPSI_AnalogMeter;                 //TB
22182: 982 unit uPSI_XPrinter,                    //TB
22183: 983 unit uPSI_IniFiles;                    //VCL
22184: 984 unit uPSI_lazIniFiles;                 //FP
22185: 985 uPSI_testutils;                        //FP
22186: 986 uPSI_ToolsUnit; (DBTests)              //FP
22187: 987 uPSI_fpcunit                           //FP
22188: 988 uPSI_testdecorator;                    //FP
22189: 989 unit uPSI_fpcunittests;                //FP
22190: 990 unit uPSI_cTCPBuffer;                  //Fundamentals 4
22191: 991 unit uPSI_Glut,                        //Open GL
22192: 992 uPSI_LEDBitmaps,                       //mX4
22193: 993 uPSI_FileClass,                        //Inno
22194: 994 uPSI_FileUtilsClass,                   //mX4
22195: 995 uPSI_ComPortInterface; //Kit          //mX4
22196: 996 unit uPSI_SwitchLed;                   //mX4
22197: 997 unit uPSI_cyDmmCanvas;                 //cY
22198: 998 uPSI_uColorFunctions;                  //DFF
22199: 999 uPSI_uSettings;                        //DFF
22200: 1000 uPSI_cyDebug.pas                      //cY
22201: 1001 uPSI_cyColorMatrix;                   //cY
22202: 1002 unit uPSI_cyCopyFiles;                //cY
22203: 1003 unit uPSI_cySearchFiles;              //cY
22204: 1004 unit uPSI_cyBaseMeasure;              //cY
22205: 1005 unit uPSI_PJIStreams;                 //DD
22206: 1006 unit uPSI_cyRunTimeResize;            //cY
22207: 1007 unit uPSI_jcontrolutils;              //cY
22208: 1008 unit uPSI_kcMapViewer; (+GEONames)    //kc
```

```
22209: 1009 unit uPSI_kcMapViewerDESynapse;           //kc
22210: 1010 unit uPSI_cparserutils; (+GIS_SysUtils)   //kc
22211: 1011 unit uPSI_LedNumber;                      //TurboPower
22212: 1012 unit uPSI_StStrL;                         //SysTools
22213:
22214:
22215: ////////////////////////////////////////////////////////////////////////////
22216: //Form Template Library FTL
22217: ////////////////////////////////////////////////////////////////////////////
22218:
22219: 32 FTL For Form Building out of the Script, eg. 399_form_templates.txt
22220:
22221: 045 unit uPSI_VListView                        TFormListView;
22222: 263 unit uPSI_JvProfiler32;                    TProfReport
22223: 270 unit uPSI_ImgList;                         TCustomImageList
22224: 278 unit uPSI_JvImageWindow;                   TJvImageWindow
22225: 317 unit uPSI_JvParserForm;                    TJvHTMLParserForm
22226: 497 unit uPSI_DebugBox;                        TDebugBox
22227: 513 unit uPSI_ImageWin;                        TImageForm, TImageForm2
22228: 514 unit uPSI_CustomDrawTreeView;             TCustomDrawForm
22229: 515 unit uPSI_GraphWin;                       TGraphWinForm
22230: 516 unit uPSI_actionMain;                     TActionForm
22231: 518 unit uPSI_CtlPanel;                       TAppletApplication
22232: 529 unit uPSI_MDIEdit;                       TEditForm
22233: 535 unit uPSI_CoolMain; {browser}            TWebMainForm
22234: 538 unit uPSI_frmExportMain;                 TSynexportForm
22235: 585 unit uPSI_usniffer; {//PortScanForm}     TSniffForm
22236: 600 unit uPSI_ThreadForm;                    TThreadSortForm;
22237: 618 unit uPSI_delphi_arduino_Unit1;          TLEDForm
22238: 620 unit uPSI_fplotMain;                     TfplotForm1
22239: 660 unit uPSI_JvDBQueryParamsForm;           TJvQueryParamsDialog
22240: 677 unit uPSI_OptionsFrm;                    TfrmOptions;
22241: 718 unit uPSI_MonForm;                       TMonitorForm
22242: 742 unit uPSI_SimplePortMain;               TPortForm1
22243: 770 unit uPSI_ovccalc;                      TOvcCalculator  //widget
22244: 810 unit uPSI_DbExcept;                     TDbEngineErrorDlg
22245: 812 unit uPSI_GL_actorUnit1;               TglActorForm1  //OpenGL Robot
22246: 846 unit uPSI_synhttp_daemon;               TTCPHttpDaemon, TTCPHttpThrd, TPingThread
22247: 867 unit uPSI_Debug;                        TfrmDebug
22248: 904 unit uPSI_U_HexView;                    THexForm2
22249: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain) TOscfrmMain
22250: 959 unit uPSI_AdSelCom;                     TComSelectForm
22251:
22252:
22253: ex.:with TEditForm.create(self) do begin
22254:       caption:= 'Template Form Tester';
22255:       FormStyle:= fsStayOnTop;
22256:       with editor do begin
22257:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
22258:         SelStart:= 0;
22259:         Modified:= False;
22260:       end;
22261:     end;
22262:   with TWebMainForm.create(self) do begin
22263:     URLs.Text:= 'http://www.kleiner.ch';
22264:     URLsClick(self); Show;
22265:   end;
22266:   with TSynexportForm.create(self) do begin
22267:     Caption:= 'Synexport HTML RTF tester';
22268:     Show;
22269:   end;
22270:   with TThreadSortForm.create(self) do begin
22271:      showmodal; free;
22272:   end;
22273:   with TCustomDrawForm.create(self) do begin
22274:        width:=820; height:=820;
22275:        image1.height:= 600; //add properties
22276:        image1.picture.bitmap:= image2.picture.bitmap;
22277:        //SelectionBackground1Click(self) CustomDraw1Click(self);
22278:        Background1.click;
22279:        bitmap1.click;
22280:        Tile1.click;
22281:        Showmodal;
22282:        Free;
22283:      end;
22284:    with TfplotForm1.Create(self) do begin
22285:      BtnPlotClick(self);
22286:      Showmodal; Free;
22287:    end;
22288:   with TOvcCalculator.create(self) do begin
22289:     parent:= aForm;
22290:    //free;
22291:     setbounds(550,510,200,150);
22292:     displaystr:= 'maXcalc';
22293:   end;
22294:   with THexForm2.Create(self) do begin
22295:      ShowModal;
22296:      Free;
22297:   end;
```

```
22298:
22299: function CheckBox: string;
22300: var idHTTP: TIDHTTP;
22301:  begin
22302:    result:= 'version not found';
22303:    if IsInternet then begin
22304:      idHTTP:= TIdHTTP.Create(NIL);
22305:      try
22306:        result:= idHTTP.Get(MXVERSIONFILE2);
22307:        result:= result[1]+result[2]+result[3]+result[4]+result[5];
22308:        if result = MBVER2 then begin
22309:          //output.Font.Style:= [fsbold];
22310:          //Speak(' A new Version '+vstr+' of max box is available! ');
22311:          result:= ('!!! OK. You have latest Version: '+result+' available at '+MXSITE);
22312:        end;
22313:       //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
22314:      finally
22315:        idHTTP.Free
22316:    end;
22317:  end;
22318: end;
22319:
22320:
22321: ////////////////////////////////////////////////////////////////////////////
22322: All maXbox Tutorials Table of Content 2014
22323: ////////////////////////////////////////////////////////////////////////////
22324:     Tutorial 00 Function-Coding  (Blix the Programmer)
22325:     Tutorial 01 Procedural-Coding
22326:     Tutorial 02 OO-Programming
22327:     Tutorial 03 Modular Coding
22328:     Tutorial 04 UML Use Case Coding
22329:     Tutorial 05 Internet Coding
22330:     Tutorial 06 Network Coding
22331:     Tutorial 07 Game Graphics Coding
22332:     Tutorial 08 Operating System Coding
22333:     Tutorial 09 Database Coding
22334:     Tutorial 10 Statistic Coding
22335:     Tutorial 11 Forms Coding
22336:     Tutorial 12 SQL DB Coding
22337:     Tutorial 13 Crypto Coding
22338:     Tutorial 14 Parallel Coding
22339:     Tutorial 15 Serial RS232 Coding
22340:     Tutorial 16 Event Driven Coding
22341:     Tutorial 17 Web Server Coding
22342:     Tutorial 18 Arduino System Coding
22343:     Tutorial 18_3 RGB LED System Coding
22344:     Tutorial 19 WinCOM /Arduino Coding
22345:     Tutorial 20 Regular Expressions RegEx
22346:     Tutorial 21 Android Coding (coming 2013)
22347:     Tutorial 22 Services Programming
22348:     Tutorial 23 Real Time Systems
22349:     Tutorial 24 Clean Code
22350:     Tutorial 25 maXbox Configuration I+II
22351:     Tutorial 26 Socket Programming with TCP
22352:     Tutorial 27 XML & TreeView
22353:     Tutorial 28 DLL Coding (available)
22354:     Tutorial 29 UML Scripting (2014)
22355:     Tutorial 30 Web of Things (2014)
22356:     Tutorial 31 Closures (2014)
22357:     Tutorial 32 SQL Firebird (coming 2014)
22358:     Tutorial 33 Oscilloscope (coming 2015)
22359:     Tutorial 34 GPS Navigation (2014)
22360:     Tutorial 35 Web Box (available)
22361:     Tutorial 36 Unit Testing (coming 2015)
22362:     Tutorial 37 API Coding (coming 2015)
22363:     Tutorial 38 3D Coding (coming 2015)
22364:     Tutorial 39 GEO Map Coding (coming 2015)
22365:     Tutorial 40 REST Coding (coming 2015)
22366:
22367:
22368: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
22369: using Docu for this file is maxbox_functions_all.pdf
22370: PEP - Pascal Education Program  Lib Lab ShellHell
22371:
22372: http://stackoverflow.com/tags/pascalscript/hot
22373: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
22374: http://sourceforge.net/projects/maXbox   #locs:51620
22375: http://sourceforge.net/apps/mediawiki/maXbox
22376: http://www.blaisepascal.eu/
22377: https://github.com/maxkleiner/maXbox3.git
22378: http://www.heise.de/download/maxbox-1176464.html
22379: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
22380: https://www.facebook.com/pages/Programming-maXbox/166844836691703
22381: http://www.softwareschule.ch/arduino_training.pdf
22382: http://www.delphiarea.com
22383:
22384: All maXbox Examples List
22385: https://github.com/maxkleiner/maXbox3/releases
22386: ********************************************************************************************
```

22387: 000_pas_baseconvert.txt
22388: 000_pas_baseconvert.txt_encrypt
22389: 000_pas_baseconvert.txt_decrypt
22390: 001_1_pas_functest - Kopie.txt
22391: 001_1_pas_functest.txt
22392: 001_1_pas_functest2.txt
22393: 001_1_pas_functest_clx2.txt
22394: 001_1_pas_functest_clx2_2.txt
22395: 001_1_pas_functest_openarray.txt
22396: 001_pas_lottogen.txt
22397: 001_pas_lottogen_template.txt
22398: 001_pas_lottogen.txtcopy
22399: 002_pas_russianroulette.txt
22400: 002_pas_russianroulette.txtcopy
22401: 002_pas_russianroulette.txtcopy_decrypt
22402: 002_pas_russianroulette.txtcopy_encrypt
22403: 003_pas_motion.txt
22404: 003_pas_motion.txtcopy
22405: 004_pas_search.txt
22406: 004_pas_search_replace.txt
22407: 004_search_replace_allfunctionlist.txt
22408: 005_pas_oodesign.txt
22409: 005_pas_shelllink.txt
22410: 006_pas_oobatch.txt
22411: 007_pas_streamcopy.txt
22412: 008_EINMALEINS_FUNC.TXT
22413: 008_explanation.txt
22414: 008_pas_verwechselt.txt
22415: 008_pas_verwechselt_ibz_bern_func.txt
22416: 008_stack_ibz.TXT
22417: 009_pas_umlrunner.txt
22418: 009_pas_umlrunner_all.txt
22419: 009_pas_umlrunner_componenttest.txt
22420: 009_pas_umlrunner_solution.txt
22421: 009_pas_umlrunner_solution_2step.txt
22422: 010_pas_oodesign_solution.txt
22423: 011_pas_puzzlepas_defect.txt
22424: 012_pas_umlrunner_solution.txt
22425: 012_pas_umlrunner_solution2.txt
22426: 013_pas_linenumber.txt
22427: 014_pas_primetest.txt
22428: 014_pas_primetest_first.txt
22429: 014_pas_primetest_sync.txt
22430: 015_pas_designbycontract.txt
22431: 015_pas_designbycontract_solution.txt
22432: 016_pas_searchrec.txt
22433: 017_chartgen.txt
22434: 018_data_simulator.txt
22435: 019_dez_to_bin.txt
22436: 019_dez_to_bin_grenzwert_ibz.txt
22437: 020_proc_feedback.txt
22438: 021_pas_symkey.txt
22439: 021_pas_symkey_solution.txt
22440: 022_pas_filestreams.txt
22441: 023_pas_find_searchrec.txt
22442: 023_pas_pathfind.txt
22443: 024_pas_TFileStream_records.txt
22444: 025_prime_direct.txt
22445: 026_pas_memorystream.txt
22446: 027_pas_shellexecute_beta.txt
22447: 027_pas_shellexecute_solution.txt
22448: 028_pas_dataset.txt
22449: 029_pas_assignfile.txt
22450: 029_pas_assignfile_dragndropexe.txt
22451: 030_palindrome_2.txt
22452: 030_palindrome_tester.txt
22453: 030_pas_recursion.txt
22454: 030_pas_recursion2.txt
22455: 031_pas_hashcode.txt
22456: 032_pas_crc_const.txt
22457: 033_pas_cipher.txt
22458: 033_pas_cipher_def.txt
22459: 033_pas_cipher_file_2_solution.txt
22460: 034_pas_soundbox.txt
22461: 035_pas_crcscript.txt
22462: 035_pas_CRCscript_modbus.txt
22463: 036_pas_includetest.txt
22464: 036_pas_includetest_basta.txt
22465: 037_pas_define_demo32.txt
22466: 038_pas_box_demonstrator.txt
22467: 039_pas_dllcall.txt
22468: 040_paspointer.txt
22469: 040_paspointer_old.txt
22470: 041_pasplotter.txt
22471: 041_pasplotter_plus.txt
22472: 042_pas_kgv_ggt.txt
22473: 043_pas_proceduretype.txt
22474: 044_pas_14queens_solwith14.txt
22475: 044_pas_8queens.txt
282_fadengraphik.txt
283_SQL_API_messagetimeout.txt
284_SysTools4.txt
285_MineForm_GR32.TXT
285_MineForm_GR32main.TXT
285_MineForm_GR32mainsolution.TXT
285_MineForm_propas.TXT
285_MineForm_propas2.TXT
285_minesweeper2.TXT
285_Patterns_process.txt
286_colormixer_jpeg_charcounter.txt
286_colormixer_jpeg_charcounter2.txt
287_eventhandling.txt
287_eventhandling2.txt
287_eventhandling2_negpower.txt
288_bitblt.txt
288_bitblt_resize.txt
289_regression.txt
289_regression2.txt
290_bestofbox.txt
290_bestofbox2.txt
290_bestofbox3.txt
291_3sort_visual_thread.txt
292_refactoring2.txt
293_bold_utils.txt
293_ib_utils.txt
293_ib_utils_timetest.txt
294_maxcalc_demo.txt
294_maxcalc_demo2.txt
295_easter_calendar.txt
295_easter_calendar2.txt
295_easter_combobox.txt
297_atomimage.txt
297_atomimage2.txt
297_atomimage3.txt
297_atomimage4.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt
299_animation.txt
299_animationmotor_arduino.txt
299_animation_formprototype.txt
299_realtimeclock_arduino.txt
299_realtimeclock_arduino2.txt
300_treeview.txt
300_treeview_test.txt
300_treeview_test2.txt
300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
309_regex_powertester3.txt

22476: 044_pas_8queens_sol2.txt
22477: 044_pas_8queens_solutions.txt
22478: 044_queens_performer.txt
22479: 044_queens_performer2.txt
22480: 044_queens_performer2tester.txt
22481: 045_pas_listhandling.txt
22482: 046_pas_records.txt
22483: 047_pas_modula10.txt
22484: 048_pas_romans.txt
22485: 049_pas_ifdemo.txt
22486: 049_pas_ifdemo_BROKER.txt
22487: 050_pas_primetest2.txt
22488: 050_pas_primetester_thieves.txt
22489: 050_program_starter.txt
22490: 050_program_starter_performance.txt
22491: 051_pas_findtext_solution.txt
22492: 052_pas_text_as_stream.txt
22493: 052_pas_text_as_stream_include.txt
22494: 053_pas_singleton.txt
22495: 054_pas_speakpassword.txt
22496: 054_pas_speakpassword2.txt
22497: 054_pas_speakpassword_searchtest.txt
22498: 055_pas_factorylist.txt
22499: 056_pas_demeter.txt
22500: 057_pas_dirfinder.txt
22501: 058_pas_filefinder.txt
22502: 058_pas_filefinder_pdf.txt
22503: 058_pas_filefinder_screview.txt
22504: 058_pas_filefinder_screview2.txt
22505: 058_pas_filefinder_screview3.txt
22506: 059_pas_timertest.txt
22507: 059_pas_timertest_2.txt
22508: 059_pas_timertest_time_solution.txt
22509: 059_timerobject_starter2.txt
22510: 059_timerobject_starter2_ibz2_async.txt
22511: 059_timerobject_starter2_uml.txt
22512: 059_timerobject_starter2_uml_main.txt
22513: 059_timerobject_starter4_ibz.txt
22514: 060_pas_datefind.txt
22515: 060_pas_datefind_exceptions2.txt
22516: 060_pas_datefind_exceptions_CHECKTEST.txt
22517: 060_pas_datefind_fulltext.txt
22518: 060_pas_datefind_plus.txt
22519: 060_pas_datefind_plus_mydate.txt
22520: 061_pas_randomwalk.txt
22521: 061_pas_randomwalk_plus.txt
22522: 062_paskorrelation.txt
22523: 063_pas_calculateform.txt
22524: 063_pas_calculateform_2list.txt
22525: 064_pas_timetest.txt
22526: 065_pas_bitcounter.txt
22527: 066_pas_eliza.txt
22528: 066_pas_eliza_include_sol.txt
22529: 067_pas_morse.txt
22530: 068_pas_piezo_sound.txt
22531: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
22532: 069_my_LEDBOX.TXT
22533: 069_pas_ledmatrix.txt
22534: 069_pas_LEDMATRIX_Alphabet.txt
22535: 069_pas_LEDMATRIX_Alphabet_run.txt
22536: 069_pas_LEDMATRIX_Alphabet_tester.txt
22537: 069_PAS_LEDMATRIX_COLOR.TXT
22538: 069_pas_ledmatrix_fixedit.txt
22539: 069_pas_LEDMATRIX_soundbox.txt
22540: 069_pas_LEDMATRIX_soundbox2.txt
22541: 069_Richter_MATRIX.TXT
22542: 070_pas_functionplot.txt
22543: 070_pas_functionplotter2.txt
22544: 070_pas_functionplotter2_mx4.txt
22545: 070_pas_functionplotter2_tester.txt
22546: 070_pas_functionplotter3.txt
22547: 070_pas_functionplotter4.txt
22548: 070_pas_functionplotter_digital.txt
22549: 070_pas_functionplotter_elliptic.txt
22550: 070_pas_function_helmholtz.txt
22551: 070_pas_textcheck_experimental.txt
22552: 071_pas_graphics.txt
22553: 071_pas_graphics_drawsym.txt
22554: 071_pas_graphics_drawsym_save.txt
22555: 071_pas_graphics_random.txt
22556: 072_pas_fractals.txt
22557: 072_pas_fractals_2.txt
22558: 072_pas_fractals_blackhole.txt
22559: 072_pas_fractals_perfomance.txt
22560: 072_pas_fractals_perfomance_new.txt
22561: 072_pas_fractals_perfomance_sharp.txt
22562: 072_pas_fractals_performance.txt
22563: 072_pas_fractals_performance_mX4.txt
22564: 073_pas_forms.txt
310_regex_decorator.TXT
312_ListView.txt
313_dmath_dll.txt
314_fundamentals4_tester.TXT
315_funcplot_dmath.TXT
316_cfileutils_cdatetime_tester.TXT
317_excel_export_tester.TXT
318_excel_export.TXT
318_excel_export2.TXT
318_excel_export3.TXT
318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
330_myclock.txt
330_myclock2.txt
331_SQL_DBfirebird4.txt
332_jprofiler.txt
332_jprofiler_form.txt
332_jprofiler_form2.txt
333_querybyexample.txt
333_querybyexample2.txt
334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docutype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_duallistview.txt
349_biginteger.txt
350_parserform.txt
351_chartform.txt
351_chartform2.txt
351_chartform3.txt
352_array_unittest.txt
353_smtp_email.txt
353_smtp_email2.txt
354_josephus.txt
355_life_of_PI.txt
356_3D_printer.txt
357_fplot.TXT
358_makesound.txt
359_charsetrules.TXT
360_allobjects.TXT
360_JvPaintFX.TXT
361_heartbeat_wave.TXT

22565: 074_pas_chartgenerator.txt
22566: 074_pas_chartgenerator_solution.txt
22567: 074_pas_chartgenerator_solution_back.txt
22568: 074_pas_charts.txt
22569: 075_bitmap_Artwork2.txt
22570: 075_pas_bitmappuzzle.txt
22571: 075_pas_bitmappuzzle24.prod.txt
22572: 075_pas_bitmappuzzle2_prod.txt
22573: 075_pas_bitmappuzzle3.txt
22574: 075_pas_bitmapsolve.txt
22575: 075_pas_bitmap_Artwork.txt
22576: 075_pas_puzzlepas_solution.txt
22577: 076_pas_3dcube.txt
22578: 076_pas_circle.txt
22579: 077_pas_mmshow.txt
22580: 078_pas_pi.txt
22581: 079_pas_3dcube_animation.txt
22582: 079_pas_3dcube_animation4.txt
22583: 079_pas_3dcube_plus.txt
22584: 080_pas_hanoi.txt
22585: 080_pas_hanoi2.txt
22586: 080_pas_hanoi2_file.txt
22587: 080_pas_hanoi2_sol.txt
22588: 080_pas_hanoi2_tester.txt
22589: 080_pas_hanoi2_tester_fast.txt
22590: 080_pas_hanoi3.txt
22591: 081_pas_chartist2.txt
22592: 082_pas_biorythmus.txt
22593: 082_pas_biorythmus_solution.txt
22594: 082_pas_biorythmus_solution_3.txt
22595: 082_pas_biorythmus_test.txt
22596: 083_pas_GITARRE.txt
22597: 083_pas_soundbox_tones.txt
22598: 084_pas_waves.txt
22599: 085_mxsinus_logo.txt
22600: 085_sinus_plot_waves.txt
22601: 086_pas_graph_arrow_heart.txt
22602: 087_bitmap_loader.txt
22603: 087_pas_bitmap_solution.txt
22604: 087_pas_bitmap_solution2.txt
22605: 087_pas_bitmap_subimage.txt
22606: 087_pas_bitmap_test.txt
22607: 088_pas_soundbox2_mp3.txt
22608: 088_pas_soundbox_mp3.txt
22609: 088_pas_sphere_2.txt
22610: 089_pas_gradient.txt
22611: 089_pas_maxland2.txt
22612: 090_pas_sudoku4.txt
22613: 090_pas_sudoku4_2.txt
22614: 091_pas_cube4.txt
22615: 092_pas_statistics4.txt
22616: 093_variance.txt
22617: 093_variance_debug.txt
22618: 094_pas_daysold.txt
22619: 094_pas_stat_date.txt
22620: 095_pas_ki_simulation.txt
22621: 096_pas_geisen_problem.txt
22622: 096_pas_montyhall_problem.txt
22623: 097_lotto_proofofconcept.txt
22624: 097_pas_lottocombinations_beat_plus.txt
22625: 097_pas_lottocombinations_beat_plus2.txt
22626: 097_pas_lottocombinations_universal.txt
22627: 097_pas_lottosimulation.txt
22628: 098_pas_chartgenerator_plus.txt
22629: 099_pas_3D_show.txt
22630: 200_big_numbers.txt
22631: 200_big_numbers2.txt
22632: 201_streamload_xml.txt
22633: 202_systemcheck.txt
22634: 203_webservice_simple_intftester.txt
22635: 204_webservice_simple.txt
22636: 205_future_value_service.txt
22637: 206_DTD_string_functions.txt
22638: 207_ibz2_async_process.txt
22639: 208_crc32_hash.txt
22640: 209_cryptohash.txt
22641: 210_public_private.txt
22642: 210_public_private_cryptosystem.txt
22643: 211_wipe_pattern.txt
22644: 211_wipe_pattern2.txt
22645: 211_wipe_pattern_solution.txt
22646: 212_pas_statisticmodule4.TXT
22647: 212_pas_statisticmoduletxt.TXT
22648: 212_statisticmodule4.txt
22649: 213_pas_BBP_Algo.txt
22650: 214_mxdocudemo.txt
22651: 214_mxdocudemo2.txt
22652: 214_mxdocudemo3.txt
22653: 215_hints_test.TXT

362_maxonmotor2.TXT
363_compress_services.txt
363_compress_services2.txt
364_pdf_services.txt
365_memorystream.txt
365_memorystream2.txt
365_memorystream_test.txt
365_U_HexView.txt
366_mp3player.txt
366_mp3player2.txt
366_mp3player2_themestest.txt
367_silvi_player_widgets.txt
367_silvi_player_widgets2.txt
367_widgets.txt
368_configuration_demo.txt
369_macro_demo.txt
370_callback2grid.TXT
370_richedit.txt
370_richedit_highlight.txt
370_synedit.txt
370_synedit2.txt
370_synedit2_mxtester.txt
370_synedit2_mxtester2.txt
371_maXbook_v4tester.txt
372stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
377_smartXMLWorkshop.txt
377_smartXMLWorkshop2.txt
378_queryperformance3.txt
378_REST1.txt
378_REST2.txt
379_timefunc.txt
379_timefunctesterfilemon.txt
380_coolfunc.txt
380_coolfunc2.txt
380_coolfunc_tester.txt
381_bitcoin_simulation.txt
382_GRMath.TXT
382_GRMath_PI_Proof.TXT
382_GRMath_Riemann.TXT
383_MDAC_DCOM.txt
384_TeamViewerID.TXT
386_InternetRadio.TXT
387_fulltextfinder.txt
387_fulltextfinder_cleancode.txt
387_fulltextfinder_fast.txt
387_fulltext_getscripttest.txt
388_TCPServerSock.TXT
388_TCPServerSock2.TXT
388_TCPServerSockClient.TXT
389_TAR_Archive.TXT
389_TAR_Archive_test.TXT
389_TAR_Archive_test2.TXT
390_Callback3.TXT
390_Callback3Rec.TXT
390_CallbackClean.TXT
390_StringlistHTML.TXT
391_ToDo_List.TXT
392_Barcode.TXT
392_Barcode2.TXT
392_Barcode23.TXT
392_Barcode2scholz.TXT
392_Barcode3scholz.TXT
393_QRCode.TXT
393_QRCode2.TXT
393_QRCode2Direct.TXT
393_QRCode2DirectIndy.TXT
393_QRCode2Direct_detlef.TXT
393_QRCode3.TXT
394_networkgraph.TXT
394_networkgraph_depwalkutilstest.TXT
394_networkgraph_depwalkutilstest2.TXT
395_USBController.TXT
396_Sort.TXT
397_Hotlog.TXT
397_Hotlog2.TXT
398_ustrings.txt
399_form_templates.txt
400_fplottchart.TXT

22654: 216_warnings_test.TXT
22655: 217_pas_heartbeat.txt
22656: 218_biorythmus_studio.txt
22657: 219_cipherbox.txt
22658: 219_crypt_source_comtest_solution.TXT
22659: 220_cipherbox_form.txt
22660: 220_cipherbox_form2.txt
22661: 221_bcd_explain.txt
22662: 222_memoform.txt
22663: 223_directorybox.txt
22664: 224_dialogs.txt
22665: 225_sprite_animation.txt
22666: 226_ASCII_Grid2.TXT
22667: 227_animation.txt
22668: 227_animation2.txt
22669: 228_android_calendar.txt
22670: 229_android_game.txt
22671: 229_android_game_tester.txt
22672: 230_DataProvider.txt
22673: 230_DataSetProvider.txt
22674: 230_DataSetXMLBackupScholz.txt
22675: 231_DBGrid_access.txt
22676: 231_DBGrid_XMLaccess.txt
22677: 231_DBGrid_XMLaccess2.txt
22678: 231_DBGrid_XMLaccess_locatetester.txt
22679: 231_DBGrid_XML_CDS_local.txt
22680: 232_outline.txt
22681: 232_outline_2.txt
22682: 233_modular_form.txt
22683: 234_debugoutform.txt
22684: 235_fastform.TXT
22685: 236_componentpower.txt
22686: 236_componentpower_back.txt
22687: 237_pas_4forms.txt
22688: 238_lottogen_form.txt
22689: 239_pas_sierpinski.txt
22690: 239_pas_sierpinski2.txt
22691: 240_unitGlobal_tester.txt
22692: 241_db3_sql_tutorial.txt
22693: 241_db3_sql_tutorial2.txt
22694: 241_db3_sql_tutorial2fix.txt
22695: 241_db3_sql_tutorial3.txt
22696: 241_db3_sql_tutorial3connect.txt
22697: 241_db3_sql_tutorial3_fptest.txt
22698: 241_RTL_SET2.txt
22699: 241_RTL_SET2_tester.txt
22700: 242_Component_Control.txt
22701: 243_tutorial_loader.txt
22702: 244_script_loader_loop.txt
22703: 245_formapp2.txt
22704: 245_formapp2_tester.txt
22705: 245_formapp2_testerX.txt
22706: 246_httpapp.txt
22707: 247_datecalendar.txt
22708: 248_ASCII_Grid2_sorted.TXT
22709: 249_picture_grid.TXT
22710: 250_tipsandtricks2.txt
22711: 250_tipsandtricks3.txt
22712: 250_tipsandtricks3api.txt
22713: 250_tipsandtricks3_admin_elevation.txt
22714: 250_tipsandtricks3_tester.txt
22715: 250_tipsandtricks4_tester.txt
22716: 250_tipsandtricks4_tester2.txt
22717: 251_compare_noise_gauss.txt
22718: 251_whitenoise.txt
22719: 251_whitenoise2.txt
22720: 252_hilbert_turtle.txt
22721: 252_pas_hilbert.txt
22722: 253_opearatingsystem3.txt
22723: 254_dynarrays.txt
22724: 255_einstein.txt
22725: 256_findconsts_of_EXE.txt
22726: 256_findfunctions2_of_EXE.txt
22727: 256_findfunctions2_of_EXEaverp.txt
22728: 256_findfunctions2_of_EXEspec.txt
22729: 256_findfunctions3.txt
22730: 256_findfunctions_of_EXE.txt
22731: 257_AES_Cipher.txt
22732: 258_AES_cryptobox.txt
22733: 258_AES_cryptobox2.txt
22734: 258_AES_cryptobox2_passdlg.txt
22735: 259_AES_crypt_directory.txt
22736: 260_sendmessage_2.TXT
22737: 260_sendmessage_beta.TXT
22738: 261_probability.txt
22739: 262_mxoutputdemo4.txt
22740: 263_async_sound.txt
22741: 264_vclutils.txt
22742: 264_VCL_utils2.txt

400_fplottchart2.TXT
400_fplottchart2teetest.TXT
400_QRCodeMarket.TXT
401_tfilerun.txt
402_richedit2.txt
403_outlookspy.txt
404_simplebrowser.txt
405_datefinder_today.txt
406_portscan.txt
407_indydemo.txt
408_testroboter.txt
409_excel_control.txt
410_keyboardevent.txt
411_json_test.txt
412_Zeosutils.txt
413_listview2.txt
414_avrdude_flash.txt
415_avrdude_writehex.txt
416_sonar_startscriptEKON.TXT
416_sonar_startscriptEKON_reporting.TXT
417_GRMath_PI_Proof2.TXT
418_functional_paradigm.txt
419_archimedes_spiral.txt
419_archimedes_spiral2.txt
420_archimedes_arduino.txt
420_Lissajous.txt
421_PI_Power.TXT
421_PI_Power2.TXT
422_world_bitboxx.txt
423_game_of_life.TXT
423_game_of_life2.TXT
423_game_of_life3.TXT
423_game_of_life3_test.TXT
423_game_of_life4.TXT
423_game_of_life4_kryptum.TXT
424_opengl_tester.txt
425_reversi_game.txt
426_IBUtils.TXT
427_IBDatabase.TXT
428_SortGrid.TXT
429_fileclass.txt
430_fileoperation.txt
430_fileoperation_tester.txt
431_performance_index.txt
432_shortstring_routines.txt
433_video_avicap.txt
433_video_avicap2.txt
434_GSM_module.TXT
435_httpcommon.txt
436_GraphicSplitter.txt
436_GraphicSplitter_form.txt
436_GraphicSplitter_form2.txt
436_teetest_screen.TXT
436_teetest_screen2.TXT
437_WinAPItop.txt
437_WinAPItop_Firebirdtester.txt
438_OvcInternational.txt
439_AsyncFreeDemo.txt
439_AsyncFreeDemoForm.txt
440_DLL_Tutor.txt
440_DLL_Tutor2.txt
440_XML_Tutor.txt
440_XML_Tutor2.txt
441_make_app.txt
442_arduino_rgb_led.txt
443_webserver_arduino_rgb_light.txt
443_webserver_arduino_rgb_light4.txt
444_webserver_arduino3ibz_rgb_led_basta.txt
445_datagrid.txt
445_datagrid2.txt
445_datagrid_android_arduino.txt
446_arduino_timer.txt
447_patternFrm_mx3.txt
448_Synapse.txt
448_Synapse2.txt
449_dweb_start_tester.txt
450_Synapse_HTTPS.txt
450_Synapse_Mime.txt
450_Synapse_ScanPing.txt
451_ocx_player.txt
451_OCX_WinPlayer2.txt
452_dbtreeview.txt
452_dbtreeview2.txt
453_stdfuncs.txt
454_fileStream.txt
455_functionfun.txt
455_functionfun2.txt
455_functionfun2_test.txt
457_ressource_grid.txt

```
22743: 265_timer_API.txt                                  458_atomimageX.txt
22744: 266_serial_interface.txt                           459_cindyfunc.txt
22745: 266_serial_interface2.txt                          459_cindyfunc2.txt
22746: 266_serial_interface3.txt                          460_TopTenFunctions.txt
22747: 267_ackermann_rec.txt                              461_sqlform_calwin.txt
22748: 267_ackermann_variants.txt                         462_caesarcipher.txt
22749: 268_DBGrid_tree.txt                                463_global_exception.txt
22750: 269_record_grid.TXT                                464_function_procedure.txt
22751: 270_Jedi_FunctionPower.txt                         464_function_procedure2.txt
22752: 270_Jedi_FunctionPowertester.txt                   464_function_procedure3.txt
22753: 271_closures_study.txt                             465_U_HexView.txt
22754: 271_closures_study_workingset2.txt                 466_moon.txt
22755: 272_pas_function_show.txt                          466_moon_inputquery.txt
22756: 273_pas_function_show2.txt                         4671_cardmagic.txt
22757: 274_library_functions.txt                          467_helmholtz_graphic.txt
22758: 275_turtle_language.txt                            468_URLMon.txt
22759: 275_turtle_language_save.txt                       468_URLMon2.txt
22760: 276_save_algo.txt                                  469_formarrow.txt
22761: 276_save_algo2.txt                                 469_formarrow_datepicker.txt
22762: 277_functionsfor39.txt                             469_formarrow_datepicker_ibz_result.txt
22763: 278_DB_Dialogs.TXT                                 469_ibzresult.txt
22764: 279_hexer2.TXT                                     470_DFFUtils_compiled.txt
22765: 279_hexer2macro.TXT                                470_DFFUtils_ScrollingLED.txt
22766: 279_hexer2macroback.TXT                            470_Oscilloscope.txt
22767: 280_UML_process.txt                                470_Oscilloscope_code.txt
22768: 280_UML_process_knabe2.txt                         471_cardmagic.txt
22769: 280_UML_process_knabe3.txt                         471_cardmagic2.txt
22770: 280_UML_process_TIM_Botzenhardt.txt                472_allcards.TXT
22771: 280_UML_TIM_Seitz.txt                              473_comboset.txt
22772: 281_picturepuzzle.txt                              474_wakeonlan.txt
22773: 281_picturepuzzle2.txt                             474_wakeonlan2.txt
22774: 281_picturepuzzle3.txt                             476_getscripttest.txt
22775: 281_picturepuzzle4.txt                             477_filenameonly.txt
22776: 479_inputquery.txt                                 480_regex_pathfinder.txt
22777: 480_regex_pathfinder2.txt                          481_processList.txt
22778: 482 processPipe.txt                                482 processPipeGCC.txt
22779: 483_PathFuncTest_mX.txt                            484_filefinder3.txt
22780: 485_InnoFunc.txt                                   486_VideoGrabber.txt
22781: 487_asyncKeyState.txt                              488_asyncTerminal.txt
22782: 489_simpleComport.txt                              490_webCamproc.txt
22783: 491_analogmeter.txt                                492_snowflake2.txt
22784: 493_gadgets.txt                                    495_fourierfreq.txt
22785: 496_InstallX.txt                                   497_LED.txt
22786: 498_UnitTesting.txt                                499_mulu42.txt
22787: 500_diceoflifes.txt                                501_firebird_datasnap_tests.txt
22788: 502_findalldocs.txt                                503_led_switch.txt
22789: 504_fileclass.txt                                  505_debug.txt
22790: 506_colormatrix.txt                                507_derutils.txt
22791: 508_simplecomportmorse.txt                         509_GEOMap2.txt
22792: 509_509_GEOMap2_SReverse.TXT                       510_510_bonn_gpsdata_mX4.pas
22793: 511_LEDLabel.txt                                   512_LED_moon.txt
22794:
22795:
22796: Web Script Examples:
22797:
22798: http://www.softwareschule.ch/examples/performer.txt';
22799: http://www.softwareschule.ch/examples/turtle.txt';
22800: http://www.softwareschule.ch/examples/SQLExport.txt';
22801: http://www.softwareschule.ch/examples/Richter.txt';
22802: http://www.softwareschule.ch/examples/checker.txt';
22803: http://www.softwareschule.ch/examples/demoscript.txt';
22804: http://www.softwareschule.ch/examples/ibzresult.txt';
22805: http://www.softwareschule.ch/examples/performindex.txt
22806: http://www.softwareschule.ch/examples/processlist.txt
22807: http://www.softwareschule.ch/examples/game.txt
22808:
22809:
22810:
22811: Delphi Basics Run Time Library listing
22812: ********************************************************************************
22813: A
22814: Compiler Directive $A Determines whether data is aligned or packed
22815: Compiler Directive $Align Determines whether data is aligned or packed
22816: Compiler Directive $AppType Determines the application type : GUI or Console
22817: Procedure SysUtils Abort Aborts the current processing with a silent exception
22818: Function System Abs Gives the absolute value of a number (-ve sign is removed)
22819: Directive Abstract Defines a class method only implemented in subclasses
22820: Variable System AbstractErrorProc Defines a proc called when an abstract method is called
22821: Function System Addr Gives the address of a variable, function or procedure
22822: Keyword And Boolean and or bitwise and of two arguments
22823: Type System AnsiChar A character type guaranteed to be 8 bits in size
22824: Function SysUtils AnsiCompareStr Compare two strings for equality
22825: Function SysUtils AnsiCompareText Compare two strings for equality, ignoring case
22826: Function StrUtils AnsiContainsStr Returns true if a string contains a substring
22827: Function StrUtils AnsiEndsStr Returns true if a string ends with a substring
22828: Function StrUtils AnsiIndexStr Compares a string with a list of strings - returns match index
22829: Function StrUtils AnsiLeftStr Extracts characters from the left of a string
22830: Function SysUtils AnsiLowerCase Change upper case characters in a string to lower case
22831: Function StrUtils AnsiMatchStr Returns true if a string exactly matches one of a list of strings
```

```
22832: Function StrUtils AnsiMidStr Returns a substring from the middle characters of a string
22833: Function StrUtils AnsiPos Find the position of one string in another
22834: Function StrUtils AnsiReplaceStr Replaces a part of one string with another
22835: Function StrUtils AnsiReverseString Reverses the sequence of letters in a string
22836: Function StrUtils AnsiRightStr Extracts characters from the right of a string
22837: Function StrUtils AnsiStartsStr Returns true if a string starts with a substring
22838: Type System AnsiString A data type that holds a string of AnsiChars
22839: Function SysUtils AnsiUpperCase Change lower case characters in a string to upper case
22840: Procedure System Append Open a text file to allow appending of text to the end
22841: Procedure SysUtils AppendStr Concatenate one string onto the end of another
22842: Function Math ArcCos The Arc Cosine of a number, returned in radians
22843: Function Math ArcSin The Arc Sine of a number, returned in radians
22844: Function System ArcTan The Arc Tangent of a number, returned in radians
22845: Keyword Array A data type holding indexable collections of data
22846: Keyword As Used for casting object references
22847: Procedure System Assign Assigns a file handle to a binary or text file
22848: Function System Assigned Returns true if a reference is not nil
22849: Procedure System AssignFile Assigns a file handle to a binary or text file
22850: Procedure Printers AssignPrn Treats the printer as a text file - an easy way of printing text
22851:
22852: B
22853: Compiler Directive $B Whether to short cut and and or operations
22854: Compiler Directive $BoolEval Whether to short cut and and or operations
22855: Procedure SysUtils Beep Make a beep sound
22856: Keyword Begin Keyword that starts a statement block
22857: Function System BeginThread Begins a separate thread of code execution
22858: Procedure System BlockRead Reads a block of data records from an untyped binary file
22859: Procedure System BlockWrite Writes a block of data records to an untyped binary file
22860: Type System Boolean Allows just True and False values
22861: Function Classes Bounds Create a TRect value from top left and size values
22862: Procedure System Break Forces a jump out of a single loop
22863: Type System Byte An integer type supporting values 0 to 255
22864:
22865: C
22866: Type System Cardinal The basic unsigned integer type
22867: Keyword Case A mechanism for acting upon different values of an Ordinal
22868: Function StdConvs CelsiusToFahrenheit Convert a celsius temperature into fahrenheit
22869: Function SysUtils ChangeFileExt Change the extension part of a file name
22870: Type System Char Variable type holding a single character
22871: Procedure System ChDir Change the working drive plus path for a specified drive
22872: Function System Chr Convert an integer into a character
22873: Keyword Class Starts the declaration of a type of object class
22874: Procedure System Close Closes an open file
22875: Procedure System CloseFile Closes an open file
22876: Variable System CmdLine Holds the execution text used to start the current program
22877: Type System Comp A 64 bit signed integer
22878: Function SysUtils CompareStr Compare two strings to see which is greater than the other
22879: Function SysUtils CompareText Compare two strings for equality, ignoring case
22880: Function Math CompareValue Compare numeric values with a tolerance
22881: Function System Concat Concatenates one or more strings into one string
22882: Keyword Const Starts the definition of fixed data values
22883: Keyword Constructor Defines the method used to create an object from a class
22884: Procedure System Continue Forces a jump to the next iteration of a loop
22885: Function ConvUtils Convert Convert one measurement value to another
22886: Function System Copy Create a copy of part of a string or an array
22887: Function System Cos The Cosine of a number
22888: Function SysUtils CreateDir Create a directory
22889: Type System Currency A floating point type with 4 decimals used for financial values
22890: Variable SysUtils CurrencyDecimals Defines decimal digit count in the Format function
22891: Variable SysUtils CurrencyFormat Defines currency string placement in curr display functions
22892: Variable SysUtils CurrencyString The currency string used in currency display functions
22893: Function SysUtils CurrToStr Convert a currency value to a string
22894: Function SysUtils CurrToStrF Convert a currency value to a string with formatting
22895:
22896: D
22897: Compiler Directive $D Determines whether application debug information is built
22898: Compiler Directive $DebugInfo Determines whether application debug information is built
22899: Compiler Directive $Define Defines a compiler directive symbol - as used by IfDef
22900: Compiler Directive $DefinitionInfo Determines whether application symbol information is built
22901: Function SysUtils Date Gives the current date
22902: Variable SysUtils DateSeparator The character used to separate display date fields
22903: Function SysUtils DateTimeToFileDate Convert a TDateTime value to a File date/time format
22904: Function SysUtils DateTimeToStr Converts TDateTime date and time values to a string
22905: Procedure SysUtils DateTimeToString Rich formatting of a TDateTime variable into a string
22906: Function SysUtils DateToStr Converts a TDateTime date value to a string
22907: Function DateUtils DayOfTheMonth Gives day of month index for a TDateTime value (ISO 8601)
22908: Function DateUtils DayOfTheWeek Gives day of week index for a TDateTime value (ISO 8601)
22909: Function DateUtils DayOfTheYear Gives the day of the year for a TDateTime value (ISO 8601)
22910: Function SysUtils DayOfWeek Gives day of week index for a TDateTime value
22911: Function DateUtils DaysBetween Gives the whole number of days between 2 dates
22912: Function DateUtils DaysInAMonth Gives the number of days in a month
22913: Function DateUtils DaysInAYear Gives the number of days in a year
22914: Function DateUtils DaySpan Gives the fractional number of days between 2 dates
22915: Procedure System Dec Decrement an ordinal variable
22916: Variable SysUtils DecimalSeparator The character used to display the decimal point
22917: Procedure SysUtils DecodeDate Extracts the year, month, day values from a TDateTime var.
22918: Procedure DateUtils DecodeDateTime Breaks a TDateTime variable into its date/time parts
22919: Procedure SysUtils DecodeTime Break a TDateTime value into individual time values
22920: Directive Default Defines default processing for a property
```

```
22921: Function Math DegToRad Convert a degrees value to radians
22922: Procedure System Delete Delete a section of characters from a string
22923: Function SysUtils DeleteFile Delete a file specified by its file name
22924: Keyword Destructor Defines the method used to destroy an object
22925: Function SysUtils DirectoryExists Returns true if the given directory exists
22926: Function SysUtils DiskFree Gives the number of free bytes on a specified drive
22927: Function SysUtils DiskSize Gives the size in bytes of a specified drive
22928: Procedure System Dispose Dispose of storage used by a pointer type variable
22929: Keyword Div Performs integer division, discarding the remainder
22930: Keyword Do Defines the start of some controlled action
22931: Type System Double A floating point type supporting about 15 digits of precision
22932: Keyword DownTo Prefixes an decremental for loop target value
22933: Function StrUtils DupeString Creates a string containing copies of a substring
22934: Directive Dynamic Allows a class method to be overriden in derived classes
22935:
22936: E
22937: Compiler Directive $Else Starts the alternate section of an IfDef or IfNDef
22938: Compiler Directive $EndIf Terminates conditional code compilation
22939: Compiler Directive $ExtendedSyntax Controls some Pascal extension handling
22940: Keyword Else Starts false section of if, case and try statements
22941: Function SysUtils EncodeDate Build a TDateTime value from year, month and day values
22942: Function DateUtils EncodeDateTime Build a TDateTime value from day and time values
22943: Function SysUtils EncodeTime Build a TDateTime value from hour, min, sec and msec values
22944: Keyword End Keyword that terminates statement blocks
22945: Function DateUtils EndOfADay Generate a TDateTime value set to the very end of a day
22946: Function DateUtils EndOfAMonth Generate a TDateTime value set to the very end of a month
22947: Procedure System EndThread Terminates a thread with an exit code
22948: Function System Eof Returns true if a file opened with Reset is at the end
22949: Function System Eoln Returns true if the current text file is pointing at a line end
22950: Procedure System Erase Erase a file
22951: Variable System ErrorAddr Sets the error address when an application terminates
22952: Keyword Except Starts the error trapping clause of a Try statement
22953: Procedure System Exclude Exclude a value in a set variable
22954: Procedure System Exit Exit abruptly from a function or procedure
22955: Variable System ExitCode Sets the return code when an application terminates
22956: Function System Exp Gives the exponent of a number
22957: Directive System Export Makes a function or procedure in a DLL externally available
22958: Type System Extended The floating point type with the highest capacity and precision
22959: Function SysUtils ExtractFileDir Extracts the dir part of a full file name
22960: Function SysUtils ExtractFileDrive Extracts the drive part of a full file name
22961: Function SysUtils ExtractFileExt Extracts the extension part of a full file name
22962: Function SysUtils ExtractFileName Extracts the name part of a full file name
22963: Function SysUtils ExtractFilePath Extracts the path part of a full file name
22964:
22965: F
22966: Function StdConvs FahrenheitToCelsius Convert a fahrenheit temperature into celsius
22967: Keyword File Defines a typed or untyped file
22968: Function SysUtils FileAge Get the last modified date/time of a file without opening it
22969: Function SysUtils FileDateToDateTime Converts a file date/time format to a TDateTime value
22970: Function SysUtils FileExists Returns true if the given file exists
22971: Function SysUtils FileGetAttr Gets the attributes of a file
22972: Variable System FileMode Defines how Reset opens a binary file
22973: Function System FilePos Gives the file position in a binary or text file
22974: Function SysUtils FileSearch Search for a file in one or more directories
22975: Function SysUtils FileSetAttr Sets the attributes of a file
22976: Function SysUtils FileSetDate Set the last modified date and time of a file
22977: Function System FileSize Gives the size in records of an open file
22978: Procedure System FillChar Fills out a section of storage with a fill character or byte value
22979: Keyword Finally Starts the unconditional code section of a Try statement
22980: Function SysUtils FindClose Closes a successful FindFirst file search
22981: Function SysUtils FindCmdLineSwitch Determine whether a certain parameter switch was passed
22982: Function SysUtils FindFirst Finds all files matching a file mask and attributes
22983: Function SysUtils FindNext Find the next file after a successful FindFirst
22984: Function SysUtils FloatToStr Convert a floating point value to a string
22985: Function SysUtils FloatToStrF Convert a floating point value to a string with formatting
22986: Procedure System Flush Flushes buffered text file data to the file
22987: Keyword For Starts a loop that executes a finite number of times
22988: Function SysUtils ForceDirectories Create a new path of directories
22989: Function SysUtils Format Rich formatting of numbers and text into a string
22990: Function SysUtils FormatCurr Rich formatting of a currency value into a string
22991: Function SysUtils FormatDateTime Rich formatting of a TDateTime variable into a string
22992: Function SysUtils FormatFloat Rich formatting of a floating point number into a string
22993: Function System Frac The fractional part of a floating point number
22994: Procedure SysUtils FreeAndNil Free memory for an object and set it to nil
22995: Procedure System FreeMem Free memory storage used by a variable
22996: Keyword System Function Defines a subroutine that returns a value
22997:
22998: G
22999: Function SysUtils GetCurrentDir Get the current directory (drive plus directory)
23000: Procedure System GetDir Get the default directory (drive plus path) for a specified drive
23001: Function System GetLastError Gives the error code of the last failing Windows API call
23002: Procedure SysUtils GetLocaleFormatSettings Gets locale values for thread-safe functions
23003: Function System GetMem Get a specified number of storage bytes
23004: Keyword Goto Forces a jump to a label, regardless of nesting
23005:
23006: H
23007: Compiler Directive $H Treat string types as AnsiString or ShortString
23008: Compiler Directive $Hints Determines whether Delphi shows compilation hints
23009: Procedure System Halt Terminates the program with an optional dialog
```

```
23010: Function System Hi Returns the hi-order byte of a (2 byte) Integer
23011: Function System High Returns the highest value of a type or variable
23012:
23013: I
23014: Compiler Directive $I Allows code in an include file to be incorporated into a Unit
23015: Compiler Directive $IfDef Executes code if a conditional symbol has been defined
23016: Compiler Directive $IfNDef Executes code if a conditional symbol has not been defined
23017: Compiler Directive $IfOpt Tests for the state of a Compiler directive
23018: Compiler Directive $Include Allows code in an include file to be incorporated into a Unit
23019: Compiler Directive $IOChecks When on, an IO operation error throws an exception
23020: Keyword If Starts a conditional expression to determine what to do next
23021: Keyword Implementation Starts the implementation (code) section of a Unit
23022: Keyword In Used to test if a value is a member of a set
23023: Procedure System Inc Increment an ordinal variable
23024: Function DateUtils IncDay Increments a TDateTime variable by + or - number of days
23025: Procedure System Include Include a value in a set variable
23026: Function DateUtils IncMillisecond Increments a TDateTime variable by + or - number of milliseconds
23027: Function DateUtils IncMinute Increments a TDateTime variable by + or - number of minutes
23028: Function SysUtils IncMonth Increments a TDateTime variable by a number of months
23029: Function DateUtils IncSecond Increments a TDateTime variable by + or - number of seconds
23030: Function DateUtils IncYear Increments a TDateTime variable by a number of years
23031: Directive Index Principally defines indexed class data properties
23032: Constant Math Infinity Floating point value of infinite size
23033: Keyword Inherited Used to call the parent class constructor or destructor method
23034: Variable System Input Defines the standard input text file
23035: Function Dialogs InputBox Display a dialog that asks for user text input, with default
23036: Function Dialogs InputQuery Display a dialog that asks for user text input
23037: Procedure System Insert Insert a string into another string
23038: Function System Int The integer part of a floating point number as a float
23039: Type System Int64 A 64 bit sized integer - the largest in Delphi
23040: Type System Integer The basic Integer type
23041: Keyword System Interface Used for Unit external definitions, and as a Class skeleton
23042: Function SysUtils IntToHex Convert an Integer into a hexadecimal string
23043: Function SysUtils IntToStr Convert an integer into a string
23044: Function System IOResult Holds the return code of the last I/O operation
23045: Keyword Is Tests whether an object is a certain class or ascendant
23046: Function Math IsInfinite Checks whether a floating point number is infinite
23047: Function SysUtils IsLeapYear Returns true if a given calendar year is a leap year
23048: Function System IsMultiThread Returns true if the code is running multiple threads
23049: Function Math IsNaN Checks to see if a floating point number holds a real number
23050:
23051: L
23052: Compiler Directive $L Determines what application debug information is built
23053: Compiler Directive $LocalSymbols Determines what application debug information is built
23054: Compiler Directive $LongStrings Treat string types as AnsiString or ShortString
23055: Function SysUtils LastDelimiter Find the last position of selected characters in a string
23056: Function System Length Return the number of elements in an array or string
23057: Function System Ln Gives the natural logarithm of a number
23058: Function System Lo Returns the low-order byte of a (2 byte) Integer
23059: Function Math Log10 Gives the log to base 10 of a number
23060: Variable SysUtils LongDateFormat Long version of the date to string format
23061: Variable SysUtils LongDayNames An array of days of the week names, starting 1 = Sunday
23062: Type System LongInt An Integer whose size is guaranteed to be 32 bits
23063: Variable SysUtils LongMonthNames An array of days of the month names, starting 1 = January
23064: Variable SysUtils LongTimeFormat Long version of the time to string format
23065: Type System LongWord A 32 bit unsigned integer
23066: Function System Low Returns the lowest value of a type or variable
23067: Function SysUtils LowerCase Change upper case characters in a string to lower case
23068:
23069: M
23070: Compiler Directive $MinEnumSize Sets the minimum storage used to hold enumerated types
23071: Function Math Max Gives the maximum of two integer values
23072: Constant System MaxInt The maximum value an Integer can have
23073: Constant System MaxLongInt The maximum value an LongInt can have
23074: Function Math Mean Gives the average for a set of numbers
23075: Function Dialogs MessageDlg Displays a message, symbol, and selectable buttons
23076: Function Dialogs MessageDlgPos Displays a message plus buttons at a given screen position
23077: Function Math Min Gives the minimum of two integer values
23078: Constant SysUtils MinsPerDay Gives the number of minutes in a day
23079: Procedure System MkDir Make a directory
23080: Keyword Mod Performs integer division, returning the remainder
23081: Constant SysUtils MonthDays Gives the number of days in a month
23082: Function DateUtils MonthOfTheYear Gives the month of the year for a TDateTime value
23083: Procedure System Move Copy bytes of data from a source to a destination
23084:
23085: N
23086: Constant Math NaN Not a real number
23087: Variable SysUtils NegCurrFormat Defines negative amount formatting in currency displays
23088: Procedure System New Create a new pointer type variable
23089: Constant System Nil A pointer value that is defined as undetermined
23090: Keyword Not Boolean Not or bitwise not of one arguments
23091: Function SysUtils Now Gives the current date and time
23092: Variable Variants Null A variable that has no value
23093:
23094: O
23095: Compiler Directive $O Determines whether Delphi optimises code when compiling
23096: Compiler Directive $Optimization Determines whether Delphi optimises code when compiling
23097: Compiler Directive $OverFlowChecks Determines whether Delphi checks integer and enum bounds
23098: Keyword System Object Allows a subroutine data type to refer to an object method
```

```
23099: Function System Odd Tests whether an integer has an odd value
23100: Keyword Of Linking keyword used in many places
23101: Keyword On Defines exception handling in a Try Except clause
23102: Keyword Or Boolean or or bitwise or of two arguments
23103: Function System Ord Provides the Ordinal value of an integer, character or enum
23104: Directive Out Identifies a routine parameter for output only
23105: Variable System Output Defines the standard output text file
23106: Directive Overload Allows 2 or more routines to have the same name
23107: Directive Override Defines a method that replaces a virtual parent class method
23108:
23109: P
23110: Keyword Packed Compacts complex data types into minimal storage
23111: Type System PAnsiChar A pointer to an AnsiChar value
23112: Type System PAnsiString Pointer to an AnsiString value
23113: Function System ParamCount Gives the number of parameters passed to the current program
23114: Function System ParamStr Returns one of the parameters used to run the current program
23115: Type System PChar A pointer to an Char value
23116: Type System PCurrency Pointer to a Currency value
23117: Type System PDateTime Pointer to a TDateTime value
23118: Type System PExtended Pointer to a Extended floating point value
23119: Function System Pi The mathematical constant
23120: Type System PInt64 Pointer to an Int64 value
23121: Function Classes Point Generates a TPoint value from X and Y values
23122: Type System Pointer Defines a general use Pointer to any memory based data
23123: Function Classes PointsEqual Compares two TPoint values for equality
23124: Function System Pos Find the position of one string in another
23125: Function System Pred Decrement an ordinal variable
23126: Function Printers Printer Returns a reference to the global Printer object
23127: Directive Private Starts the section of private data and methods in a class
23128: Keyword System Procedure Defines a subroutine that does not return a value
23129: Procedure FileCtrl ProcessPath Split a drive/path/filename string into its constituent parts
23130: Keyword System Program Defines the start of an application
23131: Function Dialogs PromptForFileName Shows a dialog allowing the user to select a file
23132: Keyword System Property Defines controlled access to class fields
23133: Directive Protected Starts a section of class private data accesible to sub-classes
23134: Type System PShortString A pointer to an ShortString value
23135: Type System PString Pointer to a String value
23136: Function Types PtInRect Tests to see if a point lies within a rectangle
23137: Directive Public Starts an externally accessible section of a class
23138: Directive Published Starts a published externally accessible section of a class
23139: Type System PVariant Pointer to a Variant value
23140: Type System PWideChar Pointer to a WideChar
23141: Type System PWideString Pointer to a WideString value
23142:
23143: Q
23144: Compiler Directive $Q Determines whether Delphi checks integer and enum bounds
23145:
23146: R
23147: Compiler Directive $R Determines whether Delphi checks array bounds
23148: Compiler Directive $RangeChecks Determines whether Delphi checks array bounds
23149: Compiler Directive $ReferenceInfo Determines whether symbol reference information is built
23150: Compiler Directive $Resource Defines a resource file to be included in the application linking
23151: Function Math RadToDeg Converts a radian value to degrees
23152: Keyword Raise Raise an exception
23153: Function System Random Generate a random floating point or integer number
23154: Procedure System Randomize Reposition the Random number generator next value
23155: Function Math RandomRange Generate a random integer number within a supplied range
23156: Variable System RandSeed Reposition the Random number generator next value
23157: Procedure System Read Read data from a binary or text file
23158: Procedure System ReadLn Read a complete line of data from a text file
23159: Type System Real A floating point type supporting about 15 digits of precision
23160: Type System Real48 The floating point type with the highest capacity and precision
23161: Procedure System ReallocMem Reallocate an existing block of storage
23162: Function DateUtils RecodeDate Change only the date part of a TDateTime variable
23163: Function DateUtils RecodeTime Change only the time part of a TDateTime variable
23164: Keyword Record A structured data type - holding fields of data
23165: Function Classes Rect Create a TRect value from 2 points or 4 coordinates
23166: Function SysUtils RemoveDir Remove a directory
23167: Procedure System Rename Rename a file
23168: Function SysUtils RenameFile Rename a file or directory
23169: Keyword Repeat Repeat statements until a ternmination condition is met
23170: Procedure SysUtils ReplaceDate Change only the date part of a TDateTime variable
23171: Procedure SysUtils ReplaceTime Change only the time part of a TDateTime variable
23172: Procedure System Reset Open a text file for reading, or binary file for read/write
23173: Variable System Result A variable used to hold the return value from a function
23174: Procedure System ReWrite Open a text or binary file for write access
23175: Procedure System RmDir Remove a directory
23176: Function System Round Rounds a floating point number to an integer
23177: Procedure System RunError Terminates the program with an error dialog
23178:
23179: S
23180: Constant SysUtils SecsPerDay Gives the number of seconds in a day
23181: Procedure System Seek Move the pointer in a binary file to a new record position
23182: Function System SeekEof Skip to the end of the current line or file
23183: Function System SeekEoln Skip to the end of the current line or file
23184: Function FileCtrl SelectDirectory Display a dialog to allow user selection of a directory
23185: Variable System Self Hidden parameter to a method - refers to the containing object
23186: Keyword Set Defines a set of up to 255 distinct values
23187: Function SysUtils SetCurrentDir Change the current directory
```

```
23188: Procedure System SetLength Changes the size of a string, or the size(s) of an array
23189: Procedure System SetString Copies characters from a buffer into a string
23190: Keyword Shl Shift an integer value left by a number of bits
23191: Variable SysUtils ShortDateFormat Compact version of the date to string format
23192: Variable SysUtils ShortDayNames An array of days of the week names, starting 1 = Sunday
23193: Type System ShortInt An integer type supporting values -128 to 127
23194: Variable SysUtils ShortMonthNames An array of days of the month names, starting 1 = Jan
23195: Type System ShortString Defines a string of up to 255 characters
23196: Variable SysUtils ShortTimeFormat Short version of the time to string format
23197: Procedure Dialogs ShowMessage Display a string in a simple dialog with an OK button
23198: Procedure Dialogs ShowMessageFmt Display formatted data in a simple dialog with an OK button
23199: Procedure Dialogs ShowMessagePos Display a string in a simple dialog at a given screen position
23200: Keyword Shr Shift an integer value right by a number of bits
23201: Function System Sin The Sine of a number
23202: Type System Single The smallest capacity and precision floating point type
23203: Function System SizeOf Gives the storage byte size of a type or variable
23204: Function System Slice Creates a slice of an array as an Open Array parameter
23205: Type System SmallInt An Integer type supporting values from -32768 to 32767
23206: Function System Sqr Gives the square of a number
23207: Function System Sqrt Gives the square root of a number
23208: Procedure System Str Converts an integer or floating point number to a string
23209: Type System String A data type that holds a string of characters
23210: Function System StringOfChar Creates a string with one character repeated many times
23211: Function SysUtils StringReplace Replace one or more substrings found within a string
23212: Function System StringToWideChar Converts a normal string into a WideChar 0 terminated buffer
23213: Function SysUtils StrScan Searches for a specific character in a constant string
23214: Function SysUtils StrToCurr Convert a number string into a currency value
23215: Function SysUtils StrToDate Converts a date string into a TDateTime value
23216: Function SysUtils StrToDateTime Converts a date+time string into a TDateTime value
23217: Function SysUtils StrToFloat Convert a number string into a floating point value
23218: Function SysUtils StrToInt Convert an integer string into an Integer value
23219: Function SysUtils StrToInt64 Convert an integer string into an Int64 value
23220: Function SysUtils StrToInt64Def Convert a string into an Int64 value with default
23221: Function SysUtils StrToIntDef Convert a string into an Integer value with default
23222: Function SysUtils StrToTime Converts a time string into a TDateTime value
23223: Function StrUtils StuffString Replaces a part of one string with another
23224: Function System Succ Increment an ordinal variable
23225: Function Math Sum Return the sum of an array of floating point values
23226:
23227: T
23228: Function Math Tan The Tangent of a number
23229: Type Classes TBits An object that can hold an infinite number of Boolean values
23230: Variable ConvUtils TConvFamily Defines a family of measurement types as used by Convert
23231: Type ConvUtils TConvType Defines a measurement type as used by Convert
23232: Type System TDateTime Data type holding a date and time value
23233: Type System Text Defines a file as a text file
23234: Type System TextFile Declares a file type for storing lines of text
23235: Type SysUtils TFloatFormat Formats for use in floating point number display functions
23236: Type SysUtils TFormatSettings A record for holding locale values for thread-safe functions
23237: Keyword Then Part of an if statement - starts the true clause
23238: Variable SysUtils ThousandSeparator The character used to display the thousands separator
23239: Keyword ThreadVar Defines variables that are given separate instances per thread
23240: Function SysUtils Time Gives the current time
23241: Variable SysUtils TimeAMString Determines AM value in DateTimeToString procedure
23242: Variable SysUtils TimePMString Determines PM value in DateTimeToString procedure
23243: Variable SysUtils TimeSeparator The character used to separate display time fields
23244: Function SysUtils TimeToStr Converts a TDateTime time value to a string
23245: Type Classes TList General purpose container of a list of objects
23246: Keyword To Prefixes an incremental for loop target value
23247: Type System TObject The base class type that is ancestor to all other classes
23248: Function DateUtils Tomorrow Gives the date tomorrow
23249: Type Dialogs TOpenDialog Displays a file selection dialog
23250: Type Types TPoint Holds X and Y integer values
23251: Type Dialogs TPrintDialog Class that creates a printer selection and control dialog
23252: Type Types TRect Holds rectangle coordinate values
23253: Type SysUtils TReplaceFlags Defines options for the StringReplace routine
23254: Function SysUtils Trim Removes leading and trailing blanks from a string
23255: Function SysUtils TrimLeft Removes leading blanks from a string
23256: Function SysUtils TrimRight Removes trailing blanks from a string
23257: Function System Trunc The integer part of a floating point number
23258: Procedure System Truncate Truncates a file size - removes all data after the current position
23259: Keyword Try Starts code that has error trapping
23260: Type Dialogs TSaveDialog Displays a dialog for selecting a save file name
23261: Type SysUtils TSearchRec Record used to hold data for FindFirst and FindNext
23262: Type Classes TStringList Holds a variable length list of strings
23263: Type SysUtils TSysCharSet Characters used by supplied string parsing functions
23264: Type System TThreadFunc Defines the function to be called by BeginThread
23265: Variable SysUtils TwoDigitYearCenturyWindow Sets the century threshold for 2 digit year string conversions
23266: Keyword Type Defines a new category of variable or process
23267:
23268:
23269: U
23270: Compiler Directive $UnDef Undefines a compiler directive symbol - as used by IfDef
23271: Keyword Unit Defines the start of a unit file - a Delphi module
23272: Keyword Until Ends a Repeat control loop
23273: Function System UpCase Convert a Char value to upper case
23274: Function SysUtils UpperCase Change lower case characters in a string to upper case
23275: Keyword Uses Declares a list of Units to be imported
23276:
```

```
23277: V
23278: Procedure System Val Converts number strings to integer and floating point values
23279: Keyword Var Starts the definition of a section of data variables
23280: Type System Variant A variable type that can hold changing data types
23281: Function Variants VarType Gives the current type of a Variant variable
23282: Constant Variants VarTypeMask Mask for the meta-type part of a Variant variable
23283: Directive Virtual Allows a class method to be overriden in derived classes
23284:
23285: W
23286: Compiler Directive $Warnings Determines whether Delphi shows compilation warnings
23287: Keyword While Repeat statements whilst a continuation condition is met
23288: Type System WideChar Variable type holding a single International character
23289: Function System WideCharToString Copies a null terminated WideChar string to a normal string
23290: Type System WideString A data type that holds a string of WideChars
23291: Keyword With A means of simplifying references to structured variables
23292: Type System Word An integer type supporting values 0 to 65535
23293: Function SysUtils WrapText Add line feeds into a string to simulate word wrap
23294: Procedure System Write Write data to a binary or text file
23295: Procedure System WriteLn Write a complete line of data to a text file
23296:
23297: X
23298: Compiler Directive $X Controls some Pascal extension handling
23299: Keyword Xor Boolean Xor or bitwise Xor of two arguments
23300:
23301: Y
23302: Compiler Directive $Y Determines whether application symbol information is built
23303: Function DateUtils Yesterday Gives the date yesterday
23304:
23305: Z
23306: Compiler Directive $Z Sets the minimum storage used to hold enumerated types
23307:
23308: mapX:
23309:  if GetMAPX('html',ExePath+'cologne2mapX.html','cathedral cologne') then
23310:     writeln('cologne map found');
23311:     GetGeoMAP('html',ExePath+AFILENAME2,'dom cologne')
23312:     writeln(GetMapXGeoReverse('XML','47.0397826','7.62914761277888'))
23313:     OpenMapX('church trier');
23314:
23315: Signature:
23316: SHA1:  maXbox3.exe 7C600FFC801FFF2AAEC83DD220202DDF5271C64D
23317: CRC32: maXbox3.exe E5450391
23318: Ref:
23319:     1. writeln(SHA1(Exepath+'\maxbox3.exe'))
23320:     2. shdig: TSHA1Digest;
23321:        shdig:= GetSHA1OfFile(false,exepath+'\maxbox3.exe');
23322:                for i:= 0 to 19 do write(BytetoHex(shdig[i]));
23323:
23324:     3. writeln(IntToHex(CRC32OfFile(exepath+'\maxbox3.exe'),4));
23325:
23326: https://www.virustotal.com/en/file/...../
23327: SHA256:  16447177cf0d176c94487347fbdf60e7c5ab7ea3801e7ee97e76d362524ce723
23328: File name:  maXbox3.exe
23329: Compilation timestamp 2014-09-15 11:02:06
23330: Entry Point 0x01141A98
23331: Number of sections 10
23332: Detection ratio:  2 / 55
23333: Analysis date:  2014-09-15 13:26:20 UTC
23334:
23335:  PE sections
23336: Name Virtual address Virtual size Raw size Entropy MD5
23337: .text 4096 18040408 18040832 6.60 46a1ece50aec6931c60ae4be887c5c7d
23338: .itext 18046976 48092 48128 6.61 1b089606c6c8d060c83c3baac638c596
23339: .data 18096128 262484 262656 5.57 36dcd5dab9fa7715a374c0a0e707bc78
23340: .bss 18362368 366904 0 0.00 d41d8cd98f00b204e9800998ecf8427e
23341: .idata 18731008 54676 54784 5.52 a6b0d3c54de337ed632a6c8971a3f373
23342: .edata 18788352 77 512 0.89 d0a2fd532dfa7b89caed1aa2027f01be
23343: .tls 18792448 208 0 0.00 d41d8cd98f00b204e9800998ecf8427e
23344: .rdata 18796544 24 512 0.27 88146ae054658ab014d87a1027e8665e
23345: .reloc 18800640 1148292 1148416 6.75 ae08cfa66591e05fd338688f4ac226f8
23346: .rsrc 19951616 3428864 3428864 5.26 1af2a9c5989016afa10ad97423fca1c8
23347:
23348: MD5 6ffa32495c823fc0cf57818cfa7d4972
23349: SHA1 7c600ffc801fff2aaec83dd220202ddf5271c64d
23350: SHA256 16447177cf0d176c94487347fbdf60e7c5ab7ea3801e7ee97e76d362524ce723
23351: ssdeep 393216:JxuIKCOvVVvF72/nzR9JaofyZ/K+f5mc:JRKC62JfYS+f
23352: authentihash 3a66f00ba5d2445ebe122a19d5bd9df490cc0671d08375bfa722be5488f1a97e
23353: imphash 5c32bd8a9083e008ac27ac5ea1657322
23354: File size 21.9 MB ( 22985728 bytes )
23355: File type Win32 EXE
23356: Magic literal
23357: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
23358: TrID  Windows ActiveX control (36.4%)
23359: Inno Setup installer (34.3%)
23360: InstallShield setup (13.4%)
23361: Win32 EXE PECompact compressed (generic) (13.0%)
23362: Win32 Executable (generic) (1.4%)
23363:
23364:   ---- bigbitbox code_cleared_checked----
```