```
 1: ********************************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9
 3: ********************************************************************************
 4:
 5: ////////////////////////////////////////////////////////////////////////////////
 6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX
 7: --------------------------------------------------------------------------------
 8:
 9: File Size of EXE:21019136 V3.9.9.95 Mai 2014 To EKON/BASTA/JAX/IBZ/SWS
10: ***************Now the Funclist*****************************************
11: Funclist Function : 12527 //10766 //10165 (7648)
12: ***************Now the Proclist*****************************************
13: Proclist Procedure Size is: 7767  //6792 //6401 4752
14: ***************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1264 //995 //
16: def head:max: maXbox7: 04.05.2014 19:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: --------------------------------------------------------------------------------
20: Funclist total Size all is: 21558! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 20970
22: ASize of EXE: 21019136 (16586240) (13511680) (13023744)
23: SHA1 Hash of maXbox 3.9.9.95: A8BE7AD5B70ECAF9797C5C9E42A3CD40E903145F
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 CheckOpen( Status : DBIResult) : Boolean
344: Function CheckPassword( const APassword : String) : Boolean
345: Function CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt
346: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
347: function CheckSynchronize(Timeout: Integer): Boolean
348: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
349: function ChrA(const a: byte): char;
350: Function ClassIDToProgID(const ClassID: TGUID): string;
351: Function ClassNameIs(const Name: string): Boolean
352: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
353: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
354: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
355: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;
356: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;
```

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

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

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

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

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

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

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

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

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

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

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

```
1310: Function GetTableNameFromQuery( const SQL : Widestring) : Widestring
1311: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1312: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1313: Function GetTasksList( const List : TStrings) : Boolean
1314: Function getTeamViewerID: string;
1315: Function GetTemplatesFolder : string
1316: Function GetText : PwideChar
1317: function GetText:PChar
1318: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1319: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1320: Function GetTextItem( const Value : string) : Longint
1321: function GETTEXTLEN:INTEGER
1322: Function GetThreadLocale: Longint; stdcall
1323: Function GetCurrentThreadID: LongWord; stdcall;
1324: Function GetTickCount : Cardinal
1325: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal) : Cardinal
1326: Function GetTicketNr : longint
1327: Function GetTime : Cardinal
1328: Function GetTime : TDateTime
1329: Function GetTimeout : Integer
1330: Function GetTimeStr: String
1331: Function GetTimeString: String
1332: Function GetTodayFiles(startdir, amask: string): TStringlist;
1333: Function getTokenCounts : integer
1334: Function GetTotalPageFileMemory : Integer
1335: Function GetTotalPhysicalMemory : Integer
1336: Function GetTotalVirtualMemory : Integer
1337: Function GetUniqueFileName( const APath, APrefix, AExt : String) : String
1338: Function GetUseNowForDate : Boolean
1339: Function GetUserDomainName( const CurUser : string) : string
1340: Function GetUserName : string
1341: Function GetUserName: string;
1342: Function GetUserObjectName( hUserObject : THandle) : string
1343: Function GetValueBitmap( Value : TBitmap) : TBitmap
1344: Function GetValueMSec : Cardinal
1345: Function GetValueStr : String
1346: Function GetVersion: int;
1347: Function GetVersionString(FileName: string): string;
1348: Function GetVisibleNode( Index : LongInt) : TOutlineNode
1349: Function GetVolumeFileSystem( const Drive : string) : string
1350: Function GetVolumeName( const Drive : string) : string
1351: Function GetVolumeSerialNumber( const Drive : string) : string
1352: Function GetWebAppServices : IWebAppServices
1353: Function GetWebRequestHandler : IWebRequestHandler
1354: Function GetWindowCaption( Wnd : HWND) : string
1355: Function GetWindowDC(hdwnd: HWND): HDC;
1356: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1357: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1358: Function GetWindowsComputerID : string
1359: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1360: Function GetWindowsFolder : string
1361: Function GetWindowsServicePackVersion : Integer
1362: Function GetWindowsServicePackVersionString : string
1363: Function GetWindowsSystemFolder : string
1364: Function GetWindowsTempFolder : string
1365: Function GetWindowsUserID : string
1366: Function GetWindowsVersion : TWindowsVersion
1367: Function GetWindowsVersionString : string
1368: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1369: Function GMTToLocalDateTime( S : string) : TDateTime
1370: Function GotoKey : Boolean
1371: Function GradToCycle( const Grads : Extended) : Extended
1372: Function GradToDeg( const Grads : Extended) : Extended
1373: Function GradToDeg( const Value : Extended) : Extended;
1374: Function GradToDeg1( const Value : Double) : Double;
1375: Function GradToDeg2( const Value : Single) : Single;
1376: Function GradToRad( const Grads : Extended) : Extended
1377: Function GradToRad( const Value : Extended) : Extended;
1378: Function GradToRad1( const Value : Double) : Double;
1379: Function GradToRad2( const Value : Single) : Single;
1380: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1381: Function GreenComponent( const Color32 : TColor32) : Integer
1382: function GUIDToString(const GUID: TGUID): string)
1383: Function HandleAllocated : Boolean
1384: function HandleAllocated: Boolean;
1385: Function HandleRequest : Boolean
1386: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1387: Function HarmonicMean( const X : TDynFloatArray) : Float
1388: Function HasAsParent( Value : TTreeNode) : Boolean
1389: Function HASCHILDDEFS : BOOLEAN
1390: Function HasCurValues : Boolean
1391: Function HasExtendCharacter( const s : UTF8String) : Boolean
1392: Function HasFormat( Format : Word) : Boolean
1393: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1394: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1395: Function HashValue(AStream: TStream): LongWord
1396: Function HashValue(AStream: TStream): Word
1397: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1398: Function HashValue1(AStream : TStream): T4x4LongWordRecord
```

```
1399: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1400: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1401: Function HashValue16( const ASrc : string) : Word;
1402: Function HashValue16stream( AStream : TStream) : Word;
1403: Function HashValue32( const ASrc : string) : LongWord;
1404: Function HashValue32Stream( AStream : TStream) : LongWord;
1405: Function HasMergeConflicts : Boolean
1406: Function hasMoreTokens : boolean
1407: Function HASPARENT : BOOLEAN
1408: function HasParent: Boolean
1409: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1410: Function HasUTF8BOM( S : TStream) : boolean;
1411: Function HasUTF8BOM1( S : AnsiString) : boolean;
1412: Function Haversine( X : Float) : Float
1413: Function Head( s : string; const subs : string; var tail : string) : string
1414: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1415: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1416: function HELPJUMP(JUMPID:STRING):BOOLEAN
1417: Function HeronianMean( const a, b : Float) : Float
1418: function HexStrToStr(Value: string): string;
1419: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1420: function HexToBin2(HexNum: string): string;
1421: Function HexToDouble( const Hex : string) : Double
1422: function HexToInt(hexnum: string): LongInt;
1423: function HexToStr(Value: string): string;
1424: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1425: function Hi(vdat: word): byte;
1426: function HiByte(W: Word): Byte)
1427: function High: Int64;
1428: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1429: function HINSTANCE: longword;
1430: function HiWord(l: DWORD): Word)
1431: function HMODULE: longword;
1432: Function HourOf( const AValue : TDateTime) : Word
1433: Function HourOfTheDay( const AValue : TDateTime) : Word
1434: Function HourOfTheMonth( const AValue : TDateTime) : Word
1435: Function HourOfTheWeek( const AValue : TDateTime) : Word
1436: Function HourOfTheYear( const AValue : TDateTime) : Word
1437: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1438: Function HourSpan( const ANow, AThen : TDateTime) : Double
1439: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1440: Function HTMLDecode( const AStr : String) : String
1441: Function HTMLEncode( const AStr : String) : String
1442: Function HTMLEscape( const Str : string) : string
1443: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1444: Function HTTPDecode( const AStr : String) : string
1445: Function HTTPEncode( const AStr : String) : string
1446: Function Hypot( const X, Y : Extended) : Extended
1447: Function IBMax( n1, n2 : Integer) : Integer
1448: Function IBMin( n1, n2 : Integer) : Integer
1449: Function IBRandomString( iLength : Integer) : String
1450: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1451: Function IBStripString( st : String; CharsToStrip : String) : String
1452: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1453: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1454: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1455: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1456: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1457: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1458:  Function RandomString( iLength : Integer) : String');
1459:  Function RandomInteger( iLow, iHigh : Integer) : Integer');
1460:  Function StripString( st : String; CharsToStrip : String) : String');
1461:  Function FormatIdentifier( Dialect : Integer; Value : String) : String');
1462:  Function FormatIdentifierValue( Dialect : Integer; Value : String) : String');
1463:  Function ExtractIdentifier( Dialect : Integer; Value : String) : String');
1464:  Function QuoteIdentifier( Dialect : Integer; Value : String) : String');
1465:  Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string;
1466:  Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String;
1467:  function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1468: Function IconToBitmap( Ico : HICON) : TBitmap
1469: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1470: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1471: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean)
1472: function IdentToColor(const Ident: string; var Color: Longint): Boolean)
1473: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1474: Function IdGetDefaultCharSet : TIdCharSet
1475: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1476: Function IdPorts2 : TStringList
1477: Function IdToMib( const Id : string) : string
1478: Function IdSHA1Hash(apath: string): string;
1479: Function IdHashSHA1(apath: string): string;
1480: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1481: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1482: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
1483: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1484: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean) : Boolean;
1485: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1486: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer) : TDateTime
1487: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
```

```
1488: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1489: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1490: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1491: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1492: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1493: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1494: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1495: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1496: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1497: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1498: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1499: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1500: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1501: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1502: Function IncludeTrailingBackslash( S : string) : string
1503: function IncludeTrailingBackslash(const S: string): string)
1504: Function IncludeTrailingPathDelimiter( const APath : string) : string
1505: Function IncludeTrailingPathDelimiter( S : string) : string
1506: function IncludeTrailingPathDelimiter(const S: string): string)
1507: Function IncludeTrailingSlash( const APath : string) : string
1508: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliSeconds : Int64) : TDateTime
1509: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64) : TDateTime
1510: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer) : TDateTime
1511: Function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1512: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64) : TDateTime
1513: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer) : TDateTime
1514: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer) : TDateTime
1515: Function IndexOf( AClass : TClass) : Integer
1516: Function IndexOf( AComponent : TComponent) : Integer
1517: Function IndexOf( AObject : TObject) : Integer
1518: Function INDEXOF( const ANAME : String) : INTEGER
1519: Function IndexOf( const DisplayName : string) : Integer
1520: Function IndexOf( const Item : TBookmarkStr) : Integer
1521: Function IndexOf( const S : WideString) : Integer
1522: Function IndexOf( const View : TJclFileMappingView) : Integer
1523: Function INDEXOF( FIELD : TFIELD) : INTEGER
1524: Function IndexOf( ID : LCID) : Integer
1525: Function INDEXOF( ITEM : TMENUITEM) : INTEGER
1526: Function IndexOf( Value : TListItem) : Integer
1527: Function IndexOf( Value : TTreeNode) : Integer
1528: function IndexOf(const S: string): Integer;
1529: Function IndexOfName( const Name : WideString) : Integer
1530: function IndexOfName(Name: string): Integer;
1531: Function IndexOfObject( AObject : TObject) : Integer
1532: function IndexofObject(AObject:tObject):Integer
1533: Function IndexOfTabAt( X, Y : Integer) : Integer
1534: Function IndexStr( const AText : string; const AValues : array of string) : Integer
1535: Function IndexText( const AText : string; const AValues : array of string) : Integer
1536: Function IndexOfInteger( AList : TStringList; Value : Variant) : Integer
1537: Function IndexOfFloat( AList : TStringList; Value : Variant) : Integer
1538: Function IndexOfDate( AList : TStringList; Value : Variant) : Integer
1539: Function IndexOfString( AList : TStringList; Value : Variant) : Integer
1540: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1541: Function IndyGetHostName : string
1542: Function IndyInterlockedDecrement( var I : Integer) : Integer
1543: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1544: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
1545: Function IndyInterlockedIncrement( var I : Integer) : Integer
1546: Function IndyLowerCase( const A1 : string) : string
1547: Function IndyStrToBool( const AString : String) : Boolean
1548: Function IndyUpperCase( const A1 : string) : string
1549: Function InitCommonControl( CC : Integer) : Boolean
1550: Function InitTempPath : string
1551: Function InMainThread : boolean
1552: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1553: Function Input: Text)
1554: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1555: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1556: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1557: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1558: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1559: Function InquireSignal( RtlSigNum : Integer) : TSignalState
1560: Function InRangeR( const A, Min, Max : Double) : Boolean
1561: function Insert( Index : Integer) : TCollectionItem
1562: Function Insert( Index : Integer) : TComboExItem
1563: Function Insert( Index : Integer) : THeaderSection
1564: Function Insert( Index : Integer) : TListItem
1565: Function Insert( Index : Integer) : TStatusPanel
1566: Function Insert( Index : Integer) : TWorkArea
1567: Function Insert( Index : LongInt; const Text : string) : LongInt
1568: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1569: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1570: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1571: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1572: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
1573: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1574: Function Instance : Longint
1575: function InstanceSize: Longint
1576: Function Int(e : Extended) : Extended;
```

```
1577: function Int64ToStr(i: Int64): String;
1578: Function IntegerToBcd( const AValue : Integer) : TBcd
1579: Function Intensity( const Color32 : TColor32) : Integer;
1580: Function Intensity( const R, G, B : Single) : Single;
1581: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
      FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1582: Function InterestRate(NPeriods:Integer;const Payment,PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime): Extended
1583: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1584: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1585: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1586: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1587: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1588: Function IntMibToStr( const Value : string) : string
1589: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1590: Function IntToBin( Value : cardinal) : string
1591: Function IntToHex( Value : Integer; Digits : Integer) : string;
1592: function IntToHex(a: integer; b: integer): string;
1593: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1594: function IntToHex64(Value: Int64; Digits: Integer): string)
1595: Function IntTo3Str( Value : Longint; separator: string) : string
1596: Function inttobool( aInt : LongInt) : Boolean
1597: function IntToStr(i: Int64): String;
1598: Function IntToStr64(Value: Int64): string)
1599: function IOResult: Integer
1600: Function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1601: Function IsAccel(VK: Word; const Str: string): Boolean
1602: Function IsAddressInNetwork( Address : String) : Boolean
1603: Function IsAdministrator : Boolean
1604: Function IsAlias( const Name : string) : Boolean
1605: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1606: Function IsASCII( const AByte : Byte) : Boolean;
1607: Function IsASCIILDH( const AByte : Byte) : Boolean;
1608: Function IsAssembly(const FileName: string): Boolean;
1609: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1610: Function IsBinary(const AChar : Char) : Boolean
1611: function IsConsole: Boolean)
1612: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1613: Function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1614: Function IsDelphiDesignMode : boolean
1615: Function IsDelphiRunning : boolean
1616: Function IsDFAState : boolean
1617: Function IsDirectory( const FileName : string) : Boolean
1618: Function IsDomain( const S : String) : Boolean
1619: function IsDragObject(Sender: TObject): Boolean;
1620: Function IsEditing : Boolean
1621: Function ISEMPTY : BOOLEAN
1622: Function IsEqual( Value : TParameters) : Boolean
1623: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1624: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1625: Function IsFirstNode : Boolean
1626: Function IsFloatZero( const X : Float) : Boolean
1627: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1628: Function IsFormOpen(const FormName: string): Boolean;
1629: Function IsFQDN( const S : String) : Boolean
1630: Function IsGrayScale : Boolean
1631: Function IsHex( AChar : Char) : Boolean;
1632: Function IsHexString(const AString: string): Boolean;
1633: Function IsHostname( const S : String) : Boolean
1634: Function IsInfinite( const AValue : Double) : Boolean
1635: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1636: Function IsInternet: boolean;
1637: Function IsLeadChar( ACh : Char) : Boolean
1638: Function IsLeapYear( Year : Word) : Boolean
1639: function IsLeapYear(Year: Word): Boolean)
1640: function IsLibrary: Boolean)
1641: Function ISLINE : BOOLEAN
1642: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1643: Function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1644: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1645: Function IsMatch( const Pattern, Text : string) : Boolean  //Grep like RegEx
1646: Function IsMainAppWindow( Wnd : HWND) : Boolean
1647: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1648: function IsMemoryManagerSet: Boolean)
1649: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1650: function IsMultiThread: Boolean)
1651: Function IsNumeric( AChar : Char) : Boolean;
1652: Function IsNumeric2( const AString : string) : Boolean;
1653: Function IsOctal( AChar : Char) : Boolean;
1654: Function IsOctalString(const AString: string) : Boolean;
1655: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1656: Function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1657: Function IsPM( const AValue : TDateTime) : Boolean
1658: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1659: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1660: Function IsPrimeRM( N : Cardinal) : Boolean  //rabin miller
1661: Function IsPrimeTD( N : Cardinal) : Boolean  //trial division
1662: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1663: Function ISqrt( const I : Smallint) : Smallint
```

```
1664: Function IsReadOnly(const Filename: string): boolean;
1665: Function IsRectEmpty( const Rect : TRect) : Boolean
1666: function IsRectEmpty(const Rect: TRect): Boolean)
1667: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1668: Function ISRIGHTTOLEFT : BOOLEAN
1669: function IsRightToLeft: Boolean
1670: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1671: Function ISSEQUENCED : BOOLEAN
1672: Function IsSystemModule( const Module : HMODULE) : Boolean
1673: Function IsSystemResourcesMeterPresent : Boolean
1674: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1675: Function IsToday( const AValue : TDateTime) : Boolean
1676: function IsToday(const AValue: TDateTime): Boolean;
1677: Function IsTopDomain( const AStr : string) : Boolean
1678: Function IsUTF8LeadByte( Lead : Char) : Boolean
1679: Function IsUTF8String( const s : UTF8String) : Boolean
1680: Function IsUTF8TrailByte( Lead : Char) : Boolean
1681: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1682: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1683: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1684: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1685: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1686: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1687: Function IsValidIdent( Ident : string) : Boolean
1688: function IsValidIdent(const Ident: string; AllowDots: Boolean): Boolean)
1689: Function IsValidIP( const S : String) : Boolean
1690: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean
1691: Function IsValidISBN( const ISBN : AnsiString) : Boolean
1692: Function IsVariantManagerSet: Boolean; //deprecated;
1693: Function IsVirtualPcGuest : Boolean;
1694: Function IsVmWareGuest : Boolean;
1695: Function IsVCLControl(Handle: HWnd): Boolean;
1696: Function IsWhiteString( const AStr : String) : Boolean
1697: Function IsWindowResponding( Wnd : HWND; Timeout : Integer) : Boolean
1698: Function IsWoW64: boolean;
1699: Function IsWin64: boolean;
1700: Function IsWow64String(var s: string): Boolean;
1701: Function IsWin64String(var s: string): Boolean;
1702: Function IsWindowsVista: boolean;
1703: Function isPowerof2(num: int64): boolean;
1704: Function powerOf2(exponent: integer): int64;
1705: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1706: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1707: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1708: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1709: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1710: Function ItemRect( Index : Integer) : TRect
1711: function ITEMRECT(INDEX:INTEGER):TRECT
1712: Function ItemWidth( Index : Integer) : Integer
1713: Function JavahashCode(val: string): Integer;
1714: Function JosephusG(n,k: integer; var graphout: string): integer;
1715: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1716: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1717: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1718: Function KeepAlive : Boolean
1719: Function KeysToShiftState(Keys: Word): TShiftState;
1720: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1721: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1722: Function KeyboardStateToShiftState: TShiftState; overload;
1723: Function Languages : TLanguages
1724: Function Last : TClass
1725: Function Last : TComponent
1726: Function Last : TObject
1727: Function LastDelimiter( Delimiters, S : string) : Integer
1728: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1729: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1730: Function Latitude2WGS84(lat: double): double;
1731: Function LCM(m,n:longint):longint;
1732: Function LCMJ( const X, Y : Cardinal) : Cardinal
1733: Function Ldexp( const X : Extended; const P : Integer) : Extended
1734: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1735: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1736: function Length: Integer;
1737: Procedure LetFileList(FileList: TStringlist; apath: string);
1738: function lengthmp3(mp3path: string):integer;
1739: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1740: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1741: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1742: function LineStart(Buffer, BufPos: PChar): PChar
1743: function LineStart(Buffer, BufPos: PChar): PChar)
1744: function ListSeparator: char;
1745: function Ln(x: Extended): Extended;
1746: Function LnXP1( const X : Extended) : Extended
1747: function Lo(vdat: word): byte;
1748: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1749: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1750: Function LoadFileAsString( const FileName : string) : string
1751: Function LoadFromFile( const FileName : string) : TBitmapLoader
```

```
1752: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1753: Function LoadPackage(const Name: string): HMODULE
1754: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1755: Function LoadStr( Ident : Integer) : string
1756: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1757: Function LoadWideStr( Ident : Integer) : WideString
1758: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1759: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1760: Function LockServer( fLock : LongBool) : HResult
1761: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1762: Function Log( const X : Extended) : Extended
1763: Function Log10( const X : Extended) : Extended
1764: Function Log2( const X : Extended) : Extended
1765: function LogBase10(X: Float): Float;
1766: Function LogBase2(X: Float): Float;
1767: Function LogBaseN(Base, X: Float): Float;
1768: Function LogN( const Base, X : Extended) : Extended
1769: Function LogOffOS : Boolean
1770: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1771: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1772: Function LongDateFormat: string;
1773: function LongTimeFormat: string;
1774: Function LongWordToFourChar( ACardinal : LongWord) : string
1775: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1776: Function LookupName( const name : string) : TInAddr
1777: Function LookupService( const service : string) : Integer
1778: function Low: Int64;
1779: Function LowerCase( S : string) : string
1780: Function Lowercase(s : AnyString) : AnyString;
1781: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1782: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1783: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1784: function MainInstance: longword
1785: function MainThreadID: longword
1786: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1787: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1788: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1789: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1790: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1791: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1792: function MakeLong(A, B: Word): Longint)
1793: Function MakeTempFilename( const APath : String) : string
1794: Function MakeValidFileName( const Str : string) : string
1795: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1796: function MakeWord(A, B: Byte): Word)
1797: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1798: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1799: Function MapValues( Mapping : string; Value : string) : string
1800: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1801: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1802: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1803: Function MaskGetFldSeparator( const EditMask : string) : Integer
1804: Function MaskGetMaskBlank( const EditMask : string) : Char
1805: Function MaskGetMaskSave( const EditMask : string) : Boolean
1806: Function MaskIntlLiteralToChar( IChar : Char) : Char
1807: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1808: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1809: Function MaskString( Mask, Value : String) : String
1810: Function Match( const sString : string) : TniRegularExpressionMatchResul
1811: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1812: Function Matches( const Filename : string) : Boolean
1813: Function MatchesMask( const Filename, Mask : string) : Boolean
1814: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1815: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1816: Function Max( AValueOne, AValueTwo : Integer) : Integer
1817: function Max(const x,y: Integer): Integer;
1818: Function Max1( const B1, B2 : Shortint) : Shortint;
1819: Function Max2( const B1, B2 : Smallint) : Smallint;
1820: Function Max3( const B1, B2 : Word) : Word;
1821: function Max3(const x,y,z: Integer): Integer;
1822: Function Max4( const B1, B2 : Integer) : Integer;
1823: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1824: Function Max6( const B1, B2 : Int64) : Int64;
1825: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1826: Function MaxFloat( const X, Y : Float) : Float
1827: Function MaxFloatArray( const B : TDynFloatArray) : Float
1828: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1829: function MaxIntValue(const Data: array of Integer):Integer)
1830: Function MaxJ( const B1, B2 : Byte) : Byte;
1831: function MaxPath: string;
1832: function MaxValue(const Data: array of Double): Double)
1833: Function MaxCalc( const Formula : string) : Extended  //math expression parser
1834: Procedure MaxCalcF( const Formula : string);   //out to console memo2
1835: function MD5(const fileName: string): string;
1836: Function Mean( const Data : array of Double) : Extended
1837: Function Median( const X : TDynFloatArray) : Float
1838: Function Memory : Pointer
1839: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1840: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
```

```
1841: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1842: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1843: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1844: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1845: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1846: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1847: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1848: Function MibToId( Mib : string) : string
1849: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1850: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1851: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1852: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1853: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1854: Procedure GetMidiOutputs( const List : TStrings)
1855: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1856: Function MIDINoteToStr( Note : TMIDINote) : string
1857: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1858: Procedure GetMidiOutputs( const List : TStrings)
1859: Procedure MidiOutCheck( Code : MMResult)
1860: Procedure MidiInCheck( Code : MMResult)
1861: Function MilliSecondOf( const AValue : TDateTime) : Word
1862: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1863: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1864: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1865: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1866: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1867: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
1868: Function MilliSecondOfTheYear( const AValue : TDateTime) : Int64
1869: Function MilliSecondsBetween( const ANow, AThen : TDateTime) : Int64
1870: Function MilliSecondSpan( const ANow, AThen : TDateTime) : Double
1871: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
1872: Function millis: int64;
1873: Function Min( AValueOne, AValueTwo : Integer) : Integer
1874: Function Min1( const B1, B2 : Shortint) : Shortint;
1875: Function Min2( const B1, B2 : Smallint) : Smallint;
1876: Function Min3( const B1, B2 : Word) : Word;
1877: Function Min4( const B1, B2 : Integer) : Integer;
1878: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1879: Function Min6( const B1, B2 : Int64) : Int64;
1880: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1881: Function MinClientRect : TRect;
1882: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1883: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1884: Function MinFloat( const X, Y : Float) : Float
1885: Function MinFloatArray( const B : TDynFloatArray) : Float
1886: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1887: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1888: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1889: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1890: Function MinIntValue( const Data : array of Integer) : Integer
1891: function MinIntValue(const Data: array of Integer):Integer)
1892: Function MinJ( const B1, B2 : Byte) : Byte;
1893: Function MinuteOf( const AValue : TDateTime) : Word
1894: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1895: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1896: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1897: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1898: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1899: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1900: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1901: Function MinValue( const Data : array of Double) : Double
1902: function MinValue(const Data: array of Double): Double)
1903: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1904: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1905: Function ModFloat( const X, Y : Float) : Float
1906: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1907: Function Modify( const Key : string; Value : Integer) : Boolean
1908: Function ModuleCacheID : Cardinal
1909: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1910: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1911: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1912: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1913: Function MonthOf( const AValue : TDateTime) : Word
1914: Function MonthOfTheYear( const AValue : TDateTime) : Word
1915: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1916: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1917: Function MonthStr( DateTime : TDateTime) : string
1918: Function MouseCoord( X, Y : Integer) : TGridCoord
1919: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1920: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1921: Function MoveNext : Boolean
1922: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1923: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1924: Function Name : string
```

```
1925: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1926: function NetworkVolume(DriveChar: Char): string
1927: Function NEWBOTTOMLINE : INTEGER
1928: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1929: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1930: Function NEWLINE : TMENUITEM
1931: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1932: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1933: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
      const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1934: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1935: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
      TMenuItem;AENABLED:BOOL): TMENUITEM
1936: Function NEWTOPLINE : INTEGER
1937: Function Next : TIdAuthWhatsNext
1938: Function NextCharIndex( S : String; Index : Integer) : Integer
1939: Function NextRecordSet : TCustomSQLDataSet
1940: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1941: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1942: Function NextToken : Char
1943: Function nextToken : WideString
1944: function NextToken:Char
1945: Function Norm( const Data : array of Double) : Extended
1946: Function NormalizeAngle( const Angle : Extended) : Extended
1947: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
1948: Function NormalizeRect( const Rect : TRect) : TRect
1949: function NormalizeRect(const Rect: TRect): TRect;
1950: Function Now : TDateTime
1951: function Now2: tDateTime
1952: Function NumProcessThreads : integer
1953: Function NumThreadCount : integer
1954: Function NthDayOfWeek( const AValue : TDateTime) : Word
1955: Function NtProductType : TNtProductType
1956: Function NtProductTypeString : string
1957: function Null: Variant;
1958: Function NullPoint : TPoint
1959: Function NullRect : TRect
1960: Function Null2Blank(aString:String):String;
1961: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
1962: Function NumIP : integer
1963: function Odd(x: Longint): boolean;
1964: Function OffsetFromUTC : TDateTime
1965: Function OffsetPoint( const P, Offset : TPoint) : TPoint
1966: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
1967: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
1968: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
1969: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
1970: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
1971: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
1972: function OpenBit:Integer
1973: Function OpenDatabase : TDatabase
1974: Function OpenDatabase( const DatabaseName : string) : TDatabase
1975: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
1976: Function OpenObject( Value : PChar) : Boolean;
1977: Function OpenObject1( Value : string) : Boolean;
1978: Function OpenSession( const SessionName : string) : TSession
1979: Function OpenVolume( const Drive : Char) : THandle
1980: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
1981: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
1982: Function OrdToBinary( const Value : Byte) : string;
1983: Function OrdToBinary1( const Value : Shortint) : string;
1984: Function OrdToBinary2( const Value : Smallint) : string;
1985: Function OrdToBinary3( const Value : Word) : string;
1986: Function OrdToBinary4( const Value : Integer) : string;
1987: Function OrdToBinary5( const Value : Cardinal) : string;
1988: Function OrdToBinary6( const Value : Int64) : string;
1989: Function OSCheck( RetVal : Boolean) : Boolean
1990: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
1991: Function OSIdentToString( const OSIdent : DWORD) : string
1992: Function Output: Text)
1993: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
1994: Function Owner : TCustomListView
1995: function Owner : TPersistent
1996: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
1997: Function PadL( pStr : String; pLth : integer) : String
1998: Function Padl(s : AnyString;I : longInt) : AnyString;
1999: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
2000: Function PadR( pStr : String; pLth : integer) : String
2001: Function Padr(s : AnyString;I : longInt) : AnyString;
2002: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
2003: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2004: Function Padz(s : AnyString;I : longInt) : AnyString;
2005: Function PaethPredictor( a, b, c : Byte) : Byte
2006: Function PARAMBYNAME( const VALUE : String) : TPARAM
2007: Function ParamByName( const Value : WideString) : TParameter
```

```
2008: Function ParamCount: Integer
2009: Function ParamsEncode( const ASrc : string) : string
2010: function ParamStr(Index: Integer): string)
2011: Function ParseDate( const DateStr : string) : TDateTime
2012: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2013: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2014: Function PathAddExtension( const Path, Extension : string) : string
2015: Function PathAddSeparator( const Path : string) : string
2016: Function PathAppend( const Path, Append : string) : string
2017: Function PathBuildRoot( const Drive : Byte) : string
2018: Function PathCanonicalize( const Path : string) : string
2019: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2020: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2021: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2022: Function PathEncode( const ASrc : string) : string
2023: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2024: Function PathExtractFileNameNoExt( const Path : string) : string
2025: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2026: Function PathGetDepth( const Path : string) : Integer
2027: Function PathGetLongName( const Path : string) : string
2028: Function PathGetLongName2( Path : string) : string
2029: Function PathGetShortName( const Path : string) : string
2030: Function PathIsAbsolute( const Path : string) : Boolean
2031: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2032: Function PathIsDiskDevice( const Path : string) : Boolean
2033: Function PathIsUNC( const Path : string) : Boolean
2034: Function PathRemoveExtension( const Path : string) : string
2035: Function PathRemoveSeparator( const Path : string) : string
2036: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2037: Function Peek : Pointer
2038: Function Peek : TObject
2039: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2040: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
2041: function Permutation(npr, k: integer): extended;
2042: function PermutationInt(npr, k: integer): Int64;
2043: Function PermutationJ( N, R : Cardinal) : Float
2044: Function Pi : Extended;
2045: Function PiE : Extended;
2046: Function PixelsToDialogsX( const Pixels : Word) : Word
2047: Function PixelsToDialogsY( const Pixels : Word) : Word
2048: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2049: Function Point( X, Y : Integer) : TPoint
2050: function Point(X, Y: Integer): TPoint)
2051: Function PointAssign( const X, Y : Integer) : TPoint
2052: Function PointDist( const P1, P2 : TPoint) : Double;
2053: function PointDist(const P1,P2: TFloatPoint): Double;
2054: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2055: function PointDist2(const P1,P2: TPoint): Double;
2056: Function PointEqual( const P1, P2 : TPoint) : Boolean
2057: Function PointIsNull( const P : TPoint) : Boolean
2058: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2059: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2060: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2061: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2062: Function Pop : Pointer
2063: Function Pop : TObject
2064: Function PopnStdDev( const Data : array of Double) : Extended
2065: Function PopnVariance( const Data : array of Double) : Extended
2066: Function PopulationVariance( const X : TDynFloatArray) : Float
2067: function Pos(SubStr, S: AnyString): Longint;
2068: Function PosEqual( const Rect : TRect) : Boolean
2069: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2070: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2071: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2072: Function Post1( AURL : string; const ASource : TStrings) : string;
2073: Function Post2( AURL : string; const ASource : TStream) : string;
2074: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2075: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2076: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2077: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2078: Function Power( const Base, Exponent : Extended) : Extended
2079: Function PowerBig(aval, n:integer): string;
2080: Function PowerIntJ( const X : Float; N : Integer) : Float;
2081: Function PowerJ( const Base, Exponent : Float) : Float;
2082: Function PowerOffOS : Boolean
2083: Function PreformatDateString( Ps : string) : string
2084: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
      FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2085: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2086: Function Printer : TPrinter
2087: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2088: Function ProcessResponse : TIdHTTPWhatsNext
2089: Function ProduceContent : string
2090: Function ProduceContentFromStream( Stream : TStream) : string
2091: Function ProduceContentFromString( const S : string) : string
2092: Function ProgIDToClassID(const ProgID: string): TGUID;
2093: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
```

```
2094: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2095: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2096: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2097: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2098: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2099: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2100: Function Push( AItem : Pointer) : Pointer
2101: Function Push( AObject : TObject) : TObject
2102: Function Put1( AURL : string; const ASource : TStream) : string;
2103: Function Pythagoras( const X, Y : Extended) : Extended
2104: Function queryDLLInterface( var queryList : TStringList) : TStringList
2105: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2106: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2107: Function queryPerformanceCounter2(mse: int64): int64;
2108: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2109: //Function QueryPerformanceFrequency(mse: int64): boolean;
2110: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2111: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2112: Procedure QueryPerformanceCounter1(var aC: Int64);
2113: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2114: Function Quote( const ACommand : String) : SmallInt
2115: Function QuotedStr( S : string) : string
2116: Function RadToCycle( const Radians : Extended) : Extended
2117: Function RadToDeg( const Radians : Extended) : Extended
2118: Function RadToDeg( const Value : Extended) : Extended;
2119: Function RadToDeg1( const Value : Double) : Double;
2120: Function RadToDeg2( const Value : Single) : Single;
2121: Function RadToGrad( const Radians : Extended) : Extended
2122: Function RadToGrad( const Value : Extended) : Extended;
2123: Function RadToGrad1( const Value : Double) : Double;
2124: Function RadToGrad2( const Value : Single) : Single;
2125: Function RandG( Mean, StdDev : Extended) : Extended
2126: function Random(const ARange: Integer): Integer;
2127: function random2(a: integer): double
2128: function RandomE: Extended;
2129: function RandomF: Extended;
2130: Function RandomFrom( const AValues : array of string) : string;
2131: Function RandomRange( const AFrom, ATo : Integer) : Integer
2132: function randSeed: longint
2133: Function RawToDataColumn( ACol : Integer) : Integer
2134: Function Read : Char
2135: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2136: function Read(Buffer:String;Count:LongInt):LongInt
2137: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2138: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2139: Function ReadCardinal( const AConvert : boolean) : Cardinal
2140: Function ReadChar : Char
2141: Function ReadClient( var Buffer, Count : Integer) : Integer
2142: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2143: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2144: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2145: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
      Boolean):Integer
2146: Function ReadInteger( const AConvert : boolean) : Integer
2147: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2148: Function ReadLn : string
2149: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2150: function Readln(question: string): string;
2151: Function ReadLnWait( AFailCount : Integer) : string
2152: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2153: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2154: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2155: Function ReadString( const ABytes : Integer) : string
2156: Function ReadString( const Section, Ident, Default : string) : string
2157: Function ReadString( Count : Integer) : string
2158: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2159: Function ReadTimeStampCounter : Int64
2160: Function RebootOS : Boolean
2161: Function Receive( ATimeOut : Integer) : TReplyStatus
2162: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2163: Function ReceiveLength : Integer
2164: Function ReceiveText : string
2165: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2166: Function ReceiveSerialText: string
2167: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2168: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
      AMilliSec:Word):TDateTime
2169: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2170: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2171: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2172: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2173: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2174: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2175: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2176: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2177: Function Reconcile( const Results : OleVariant) : Boolean
2178: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
```

```
2179: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2180: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2181: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2182: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2183: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2184: Function RectCenter( const R : TRect) : TPoint
2185: Function RectEqual( const R1, R2 : TRect) : Boolean
2186: Function RectHeight( const R : TRect) : Integer
2187: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2188: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2189: Function RectIntersection( const R1, R2 : TRect) : TRect
2190: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2191: Function RectIsEmpty( const R : TRect) : Boolean
2192: Function RectIsNull( const R : TRect) : Boolean
2193: Function RectIsSquare( const R : TRect) : Boolean
2194: Function RectIsValid( const R : TRect) : Boolean
2195: Function RectsAreValid( R : array of TRect) : Boolean
2196: Function RectUnion( const R1, R2 : TRect) : TRect
2197: Function RectWidth( const R : TRect) : Integer
2198: Function RedComponent( const Color32 : TColor32) : Integer
2199: Function Refresh : Boolean
2200: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2201: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2202: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2203: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2204: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2205: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2206: Function ReleaseHandle : HBITMAP
2207: Function ReleaseHandle : HENHMETAFILE
2208: Function ReleaseHandle : HICON
2209: Function ReleasePalette : HPALETTE
2210: Function RemainderFloat( const X, Y : Float) : Float
2211: Function Remove( AClass : TClass) : Integer
2212: Function Remove( AComponent : TComponent) : Integer
2213: Function Remove( AItem : Integer) : Integer
2214: Function Remove( AItem : Pointer) : Pointer
2215: Function Remove( AItem : TObject) : TObject
2216: Function Remove( AObject : TObject) : Integer
2217: Function RemoveBackslash( const PathName : string) : string
2218: Function RemoveDF( aString : String) : String  //removes thousand separator
2219: Function RemoveDir( Dir : string) : Boolean
2220: function RemoveDir(const Dir: string): Boolean)
2221: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2222: Function RemoveFileExt( const FileName : string) : string
2223: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2224: Function RenameFile( OldName, NewName : string) : Boolean
2225: function RenameFile(const OldName: string; const NewName: string): Boolean)
2226: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2227: Function ReplaceText( const AText, AFromText, AToText : string) : string
2228: Function Replicate(c : char;I : longInt) : String;
2229: Function Request : TWebRequest
2230: Function ResemblesText( const AText, AOther : string) : Boolean
2231: Function Reset : Boolean
2232: function Reset2(mypath: string):string;
2233: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2234: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2235: Function Response : TWebResponse
2236: Function ResumeSupported : Boolean
2237: Function RETHINKHOTKEYS : BOOLEAN
2238: Function RETHINKLINES : BOOLEAN
2239: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2240: Function RetrieveCurrentDir : string
2241: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2242: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2243: Function RetrieveMailBoxSize : integer
2244: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2245: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2246: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2247: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2248: Function ReverseBits( Value : Byte) : Byte;
2249: Function ReverseBits1( Value : Shortint) : Shortint;
2250: Function ReverseBits2( Value : Smallint) : Smallint;
2251: Function ReverseBits3( Value : Word) : Word;
2252: Function ReverseBits4( Value : Cardinal) : Cardinal;
2253: Function ReverseBits4( Value : Integer) : Integer;
2254: Function ReverseBits5( Value : Int64) : Int64;
2255: Function ReverseBytes( Value : Word) : Word;
2256: Function ReverseBytes1( Value : Smallint) : Smallint;
2257: Function ReverseBytes2( Value : Integer) : Integer;
2258: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2259: Function ReverseBytes4( Value : Int64) : Int64;
2260: Function ReverseString( const AText : string) : string
2261: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var
      HostName:String):Bool;
2262: Function Revert : HResult
2263: Function RGB(R,G,B: Byte): TColor;
2264: Function RGB2BGR( const Color : TColor) : TColor
2265: Function RGB2TColor( R, G, B : Byte) : TColor
2266: Function RGBToWebColorName( RGB : Integer) : string
```

```
2267: Function RGBToWebColorStr( RGB : Integer) : string
2268: Function RgbToHtml( Value : TColor) : string
2269: Function HtmlToRgb(const Value: string): TColor;
2270: Function RightStr( const AStr : String; Len : Integer) : String
2271: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2272: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2273: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2274: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2275: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2276: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2277: Function Round(e : Extended) : Longint;
2278: Function Round64(e: extended): Int64;
2279: Function RoundAt( const Value : string; Position : SmallInt) : string
2280: Function RoundFrequency( const Frequency : Integer) : Integer
2281: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2282: Function RoundPoint( const X, Y : Double) : TPoint
2283: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2284: Function RowCount : Integer
2285: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2286: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2287: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2288: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2289: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2290: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2291: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2292: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2293: Function S_AddBackSlash( const ADirName : string) : string
2294: Function S_AllTrim( const cStr : string) : string
2295: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2296: Function S_Cut( const cStr : string; const iLen : integer) : string
2297: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2298: Function S_DirExists( const ADir : string) : Boolean
2299: Function S_Empty( const cStr : string) : boolean
2300: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2301: Function S_LargeFontsActive : Boolean
2302: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2303: Function S_LTrim( const cStr : string) : string
2304: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2305: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2306: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2307: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2308: Function S_RTrim( const cStr : string) : string
2309: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2310: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2311: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2312: Function S_Space( const iLen : integer) : String
2313: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2314: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2315: Function S_StrCRC32( const Text : string) : LongWORD
2316: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2317: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2318: Function S_StringtoUTF_8( const AString : string) : string
2319: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2320: function S_StrToReal(const cStr: string; var R: Double): Boolean
2321: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2322: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2323: Function S_UTF_8ToString( const AString : string) : string
2324: Function S_WBox( const AText : string) : integer
2325: Function SameDate( const A, B : TDateTime) : Boolean
2326: function SameDate(const A, B: TDateTime): Boolean;
2327: Function SameDateTime( const A, B : TDateTime) : Boolean
2328: function SameDateTime(const A, B: TDateTime): Boolean;
2329: Function SameFileName( S1, S2 : string) : Boolean
2330: Function SameText( S1, S2 : string) : Boolean
2331: function SameText(const S1: string; const S2: string): Boolean)
2332: Function SameTime( const A, B : TDateTime) : Boolean
2333: function SameTime(const A, B: TDateTime): Boolean;
2334: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2335: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2336: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2337: Function SampleVariance( const X : TDynFloatArray) : Float
2338: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2339: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2340: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2341: Function SaveToFile( const AFileName : TFileName) : Boolean
2342: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2343: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2344: Function ScanF(const aformat: String; const args: array of const): string;
2345: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2346: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
     TStringSearchOptions):PChar
2347: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
     String;Options:TStringSearchOptions):Integer;
2348: function SearchRecattr: integer;
2349: function SearchRecExcludeAttr: integer;
2350: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2351: function SearchRecname: string;
2352: function SearchRecsize: integer;
2353: function SearchRecTime: integer;
```

```
2354: Function Sec( const X : Extended) : Extended
2355: Function Secant( const X : Extended) : Extended
2356: Function SecH( const X : Extended) : Extended
2357: Function SecondOf( const AValue : TDateTime) : Word
2358: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2359: Function SecondOfTheHour( const AValue : TDateTime) : Word
2360: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2361: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2362: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2363: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2364: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2365: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2366: Function SectionExists( const Section : string) : Boolean
2367: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2368: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2369: function Seek(Offset:LongInt;Origin:Word):LongInt
2370: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2371: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2372: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2373: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2374: Function SendBuf( var Buf, Count : Integer) : Integer
2375: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2376: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2377: Function SendKey( AppName : string; Key : Char) : Boolean
2378: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2379: Function SendStream( AStream : TStream) : Boolean
2380: Function SendStreamThenDrop( AStream : TStream) : Boolean
2381: Function SendText( const S : string) : Integer
2382: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2383: Function SendSerialText(Data: String): cardinal
2384: Function Sent : Boolean
2385: Function ServicesFilePath: string
2386: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2387: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2388: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2389: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2390: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2391: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2392: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2393: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2394: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2395: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2396: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2397: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2398: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2399: Function SetCurrentDir( Dir : string) : Boolean
2400: function SetCurrentDir(const Dir: string): Boolean)
2401: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2402: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2403: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2404: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2405: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2406: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2407: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2408: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2409: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2410: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2411: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2412: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2413: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2414: Function SetLocalTime( Value : TDateTime) : boolean
2415: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2416: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2417: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2418: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2419: Function SetSize( libNewSize : Longint) : HResult
2420: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2421: Function Sgn( const X : Extended) : Integer
2422: function SHA1(const fileName: string): string;
2423: function SHA256(astr: string; amode: char): string)
2424: function SHA512(astr: string; amode: char): string)
2425: Function ShareMemoryManager : Boolean
2426: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2427: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2428: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2429: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2430: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2431: function ShortDateFormat: string;
2432: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2433: function ShortTimeFormat: string;
2434: function SHOWMODAL:INTEGER
2435: function ShowWindow(C1: HWND; C2: integer): boolean;
2436: procedure ShowMemory     //in Dialog
2437: function ShowMemory2: string;
2438: Function ShutDownOS : Boolean
2439: Function Signe( const X, Y : Extended) : Extended
2440: Function Sign( const X : Extended) : Integer
```

```
2441: Function Sin(e : Extended) : Extended;
2442: Function sinc( const x : Double) : Double
2443: Function SinJ( X : Float) : Float
2444: Function Size( const AFileName : String) : Integer
2445: function SizeOf: Longint;
2446: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2447: function SlashSep(const Path, S: String): String
2448: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2449: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2450: Function SmallPoint(X, Y: Integer): TSmallPoint)
2451: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2452: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2453: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2454: Function SoundexProc( const AText, AOther : string) : Boolean
2455: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2456: Function SoundexWord( const AText : string) : Word
2457: Function SourcePos : Longint
2458: function SourcePos:LongInt
2459: Function Split0( Str : string; const substr : string) : TStringList
2460: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2461: Function SQLRequiresParams( const SQL : WideString) : Boolean
2462: Function Sqr(e : Extended) : Extended;
2463: Function Sqrt(e : Extended) : Extended;
2464: Function StartIP : String
2465: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2466: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2467: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2468: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2469: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2470: Function StartOfAYear( const AYear : Word) : TDateTime
2471: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2472: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2473: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2474: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2475: Function StartsStr( const ASubText, AText : string) : Boolean
2476: Function StartsText( const ASubText, AText : string) : Boolean
2477: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2478: Function StartsWith( const str : string; const sub : string) : Boolean
2479: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2480: Function StatusString( StatusCode : Integer) : string
2481: Function StdDev( const Data : array of Double) : Extended
2482: Function Stop : Float
2483: Function StopCount( var Counter : TJclCounter) : Float
2484: Function StoreColumns : Boolean
2485: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2486: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2487: Function StrAlloc( Size : Cardinal) : PChar
2488: function StrAlloc(Size: Cardinal): PChar)
2489: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2490: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2491: Function StrBufSize( Str : PChar) : Cardinal
2492: function StrBufSize(const Str: PChar): Cardinal)
2493: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2494: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2495: Function StrCat( Dest : PChar; Source : PChar) : PChar
2496: function StrCat(Dest: PChar; const Source: PChar): PChar)
2497: Function StrCharLength( Str : PChar) : Integer
2498: Function StrComp( Str1, Str2 : PChar) : Integer
2499: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2500: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2501: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2502: Function Stream_to_AnsiString( Source : TStream) : ansistring
2503: Function Stream_to_Base64( Source : TStream) : ansistring
2504: Function Stream_to_decimalbytes( Source : TStream) : string
2505: Function Stream2WideString( oStream : TStream) : WideString
2506: Function StreamtoAnsiString( Source : TStream) : ansistring
2507: Function StreamToByte( Source : TStream) : string
2508: Function StreamToDecimalbytes( Source : TStream) : string
2509: Function StreamtoOrd( Source : TStream) : string
2510: Function StreamToString( Source : TStream) : string
2511: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2512: Function StrEmpty( const sString : string) : boolean
2513: Function StrEnd( Str : PChar) : PChar
2514: function StrEnd(const Str: PChar): PChar)
2515: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2516: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2517: Function StrGet(var S : String; I : Integer) : Char;
2518: Function StrGet2(S : String; I : Integer) : Char;
2519: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2520: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2521: Function StrHtmlDecode( const AStr : String) : String
2522: Function StrHtmlEncode( const AStr : String) : String
2523: Function StrToBytes(const Value: String): TBytes;
2524: Function StrIComp( Str1, Str2 : PChar) : Integer
2525: Function StringOfChar(c : char;I : longInt) : String;
2526: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2527: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2528: Function StringRefCount(const s: String): integer;
2529: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
```

```
2530: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2531: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2532: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2533: Function StringToBoolean( const Ps : string) : Boolean
2534: function StringToColor(const S: string): TColor)
2535: function StringToCursor(const S: string): TCursor;
2536: function StringToGUID(const S: string): TGUID)
2537: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2538: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2539: Function StringWidth( S : string) : Integer
2540: Function StrInternetToDateTime( Value : string) : TDateTime
2541: Function StrIsDateTime( const Ps : string) : Boolean
2542: Function StrIsFloatMoney( const Ps : string) : Boolean
2543: Function StrIsInteger( const S : string) : Boolean
2544: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2545: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2546: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2547: Function StrLen( Str : PChar) : Cardinal
2548: function StrLen(const Str: PChar): Cardinal)
2549: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2550: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2551: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2552: Function StrLower( Str : PChar) : PChar
2553: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2554: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2555: Function StrNew( Str : PChar) : PChar
2556: function StrNew(const Str: PChar): PChar)
2557: Function StrNextChar( Str : PChar) : PChar
2558: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2559: Function StrParse( var sString : string; const sDelimiters : string) : string;
2560: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2561: Function StrPas( Str : PChar) : string
2562: function StrPas(const Str: PChar): string)
2563: Function StrPCopy( Dest : PChar; Source : string) : PChar
2564: function StrPCopy(Dest: PChar; const Source: string): PChar)
2565: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2566: Function StrPos( Str1, Str2 : PChar) : PChar
2567: Function StrScan(const Str: PChar; Chr: Char): PChar)
2568: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2569: Function StrToBcd( const AValue : string) : TBcd
2570: Function StrToBool( S : string) : Boolean
2571: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2572: Function StrToCard( const AStr : String) : Cardinal
2573: Function StrToConv( AText : string; out AType : TConvType) : Double
2574: Function StrToCurr( S : string) : Currency;
2575: function StrToCurr(const S: string): Currency)
2576: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2577: Function StrToDate( S : string) : TDateTime;
2578: function StrToDate(const s: string): TDateTime;
2579: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2580: Function StrToDateTime( S : string) : TDateTime;
2581: function StrToDateTime(const S: string): TDateTime)
2582: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2583: Function StrToDay( const ADay : string) : Byte
2584: Function StrToFloat( S : string) : Extended;
2585: function StrToFloat(s: String): Extended;
2586: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2587: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2588: Function StrToFloat( S : string) : Extended;
2589: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2590: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2591: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2592: Function StrToCurr( S : string) : Currency;
2593: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2594: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2595: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2596: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2597: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2598: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2599: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2600: Function StrToDateTime( S : string) : TDateTime;
2601: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2602: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2603: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2604: Function StrToInt( S : string) : Integer
2605: function StrToInt(s: String): Longint;
2606: Function StrToInt64( S : string) : Int64
2607: function StrToInt64(s: String): int64;
2608: Function StrToInt64Def( S : string; Default : Int64) : Int64
2609: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2610: Function StrToIntDef( S : string; Default : Integer) : Integer
2611: function StrToIntDef(const S: string; Default: Integer): Integer)
2612: function StrToIntDef(s: String; def: Longint): Longint;
2613: Function StrToMonth( const AMonth : string) : Byte
2614: Function StrToTime( S : string) : TDateTime;
2615: function StrToTime(const S: string): TDateTime)
2616: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2617: Function StrToWord( const Value : String) : Word
2618: Function StrToXmlDate( const DateStr : string; const Format : string) : string
```

```
2619: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2620: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2621: Function StrUpper( Str : PChar) : PChar
2622: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2623: Function Sum( const Data : array of Double) : Extended
2624: Function SumFloatArray( const B : TDynFloatArray) : Float
2625: Function SumInt( const Data : array of Integer) : Integer
2626: Function SumOfSquares( const Data : array of Double) : Extended
2627: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2628: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2629: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2630: Function Supports( CursorOptions : TCursorOptions) : Boolean
2631: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2632: Function SwapWord(w : word): word)
2633: Function SwapInt(i : integer): integer)
2634: Function SwapLong(L : longint): longint)
2635: Function Swap(i : integer): integer)
2636: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2637: Function SyncTime : Boolean
2638: Function SysErrorMessage( ErrorCode : Integer) : string
2639: function SysErrorMessage(ErrorCode: Integer): string)
2640: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2641: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2642: Function SysStringLen(const S: WideString): Integer; stdcall;
2643: Function TabRect( Index : Integer) : TRect
2644: Function Tan( const X : Extended) : Extended
2645: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2646: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2647: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2648: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2649: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2650: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2651: Function TenToY( const Y : Float) : Float
2652: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2653: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2654: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2655: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2656: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2657: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2658: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2659: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2660: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2661: Function TestBits( const Value, Mask : Byte) : Boolean;
2662: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2663: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2664: Function TestBits3( const Value, Mask : Word) : Boolean;
2665: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2666: Function TestBits5( const Value, Mask : Integer) : Boolean;
2667: Function TestBits6( const Value, Mask : Int64) : Boolean;
2668: Function TestFDIVInstruction : Boolean
2669: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2670: Function TextExtent( const Text : string) : TSize
2671: function TextHeight(Text: string): Integer;
2672: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2673: Function TextStartsWith( const S, SubS : string) : Boolean
2674: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2675: Function ConvInteger(i : integer):string;
2676: Function IntegerToText(i : integer):string;
2677: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2678: function TextWidth(Text: string): Integer;
2679: Function ThreadCount : integer
2680: function ThousandSeparator: char;
2681: Function Ticks : Cardinal
2682: Function Time : TDateTime
2683: function Time: TDateTime;
2684: function TimeGetTime: int64;
2685: Function TimeOf( const AValue : TDateTime) : TDateTime
2686: function TimeSeparator: char;
2687: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2688: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2689: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2690: Function TimeToStr( DateTime : TDateTime) : string;
2691: function TimeToStr(const DateTime: TDateTime): string;
2692: Function TimeZoneBias : TDateTime
2693: Function ToCommon( const AValue : Double) : Double
2694: function ToCommon(const AValue: Double): Double;
2695: Function Today : TDateTime
2696: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2697: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2698: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2699: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2700: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2701: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
```

```
2702: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2703: function TokenComponentIdent:String
2704: Function TokenFloat : Extended
2705: function TokenFloat:Extended
2706: Function TokenInt : Longint
2707: function TokenInt:LongInt
2708: Function TokenString : string
2709: function TokenString:String
2710: Function TokenSymbolIs( const S : string) : Boolean
2711: function TokenSymbolIs(S:String):Boolean
2712: Function Tomorrow : TDateTime
2713: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2714: Function ToString : string
2715: Function TotalVariance( const Data : array of Double) : Extended
2716: Function Trace2( AURL : string) : string;
2717: Function TrackMenu( Button : TToolButton) : Boolean
2718: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2719: Function TranslateURI( const URI : string) : string
2720: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2721: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
      SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2722: Function Trim( S : string) : string;
2723: Function Trim( S : WideString) : WideString;
2724: Function Trim(s : AnyString) : AnyString;
2725: Function TrimAllOf( ATrim, AText : String) : String
2726: Function TrimLeft( S : string) : string;
2727: Function TrimLeft( S : WideString) : WideString;
2728: Function TrimLeft(const S: string): string)
2729: Function TrimRight( S : string) : string;
2730: Function TrimRight( S : WideString) : WideString;
2731: function TrimRight(const S: string): string)
2732: function TrueBoolStrs: array of string
2733: Function Trunc(e : Extended) : Longint;
2734: Function Trunc64(e: extended): Int64;
2735: Function TruncPower( const Base, Exponent : Float) : Float
2736: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2737: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2738: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2739: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2740: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2741: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
      AValue:TDateTime):Boolean
2742: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2743: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
      AVal:TDateTime):Bool
2744: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2745: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2746: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2747: Function TryLock : Boolean
2748: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2749: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
2750: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2751: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2752: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2753: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2754: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2755: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2756: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2757: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2758: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2759: Function TwoToY( const Y : Float) : Float
2760: Function UCS4StringToWideString( const S : UCS4String) : WideString
2761: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2762: function Unassigned: Variant;
2763: Function UndoLastChange( FollowChange : Boolean) : Boolean
2764: function UniCodeToStr(Value: string): string;
2765: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2766: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2767: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2768: Function UnixPathToDosPath( const Path : string) : string
2769: Function UnixToDateTime( const AValue : Int64) : TDateTime
2770: function UnixToDateTime(U: Int64): TDateTime;
2771: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2772: Function UnlockResource( ResData : HGLOBAL) : LongBool
2773: Function UnlockVolume( var Handle : THandle) : Boolean
2774: Function UnMaskString( Mask, Value : String) : String
2775: function UpCase(ch : Char ) : Char;
2776: Function UpCaseFirst( const AStr : string) : string
2777: Function UpCaseFirstWord( const AStr : string) : string
2778: Function UpdateAction( Action : TBasicAction) : Boolean
2779: Function UpdateKind : TUpdateKind
2780: Function UPDATESTATUS : TUPDATESTATUS
2781: Function UpperCase( S : string) : string
2782: Function Uppercase(s : AnyString) : AnyString;
2783: Function URLDecode( ASrc : string) : string
2784: Function URLEncode( const ASrc : string) : string
2785: Function UseRightToLeftAlignment : Boolean
2786: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
```

```
2787: Function UseRightToLeftReading : Boolean
2788: Function UTF8CharLength( Lead : Char) : Integer
2789: Function UTF8CharSize( Lead : Char) : Integer
2790: Function UTF8Decode( const S : UTF8String) : WideString
2791: Function UTF8Encode( const WS : WideString) : UTF8String
2792: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2793: Function Utf8ToAnsi( const S : UTF8String) : string
2794: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2795: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2796: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2797: Function ValidParentForm(control: TControl): TForm
2798: Function Value : Variant
2799: Function ValueExists( const Section, Ident : string) : Boolean
2800: Function ValueOf( const Key : string) : Integer
2801: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2802: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2803: Function VarArrayFromStrings( Strings : TStrings) : Variant
2804: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2805: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2806: Function VarFMTBcd : TVarType
2807: Function VarFMTBcdCreate1 : Variant;
2808: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2809: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2810: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2811: Function Variance( const Data : array of Double) : Extended
2812: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2813: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2814: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2815: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2816: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2817: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2818: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2819: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2820: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2821: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2822: Function VariantNeg( const V1 : Variant) : Variant
2823: Function VariantNot( const V1 : Variant) : Variant
2824: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2825: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2826: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2827: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2828: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2829: function VarIsEmpty(const V: Variant): Boolean;
2830: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2831: function VarIsNull(const V: Variant): Boolean;
2832: Function VarToBcd( const AValue : Variant) : TBcd
2833: function VarType(const V: Variant): TVarType;
2834: Function VarType( const V : Variant) : TVarType
2835: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2836: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2837: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2838: Function VarIsByRef( const V : Variant) : Boolean
2839: Function VarIsEmpty( const V : Variant) : Boolean
2840: Procedure VarCheckEmpty( const V : Variant)
2841: Function VarIsNull( const V : Variant) : Boolean
2842: Function VarIsClear( const V : Variant) : Boolean
2843: Function VarIsCustom( const V : Variant) : Boolean
2844: Function VarIsOrdinal( const V : Variant) : Boolean
2845: Function VarIsFloat( const V : Variant) : Boolean
2846: Function VarIsNumeric( const V : Variant) : Boolean
2847: Function VarIsStr( const V : Variant) : Boolean
2848: Function VarToStr( const V : Variant) : string
2849: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2850: Function VarToWideStr( const V : Variant) : WideString
2851: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2852: Function VarToDateTime( const V : Variant) : TDateTime
2853: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2854: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2855: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2856: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2857: Function VarSameValue( const A, B : Variant) : Boolean
2858: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2859: Function VarIsEmptyParam( const V : Variant) : Boolean
2860: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2861: Function VarIsError1( const V : Variant) : Boolean;
2862: Function VarAsError( AResult : HRESULT) : Variant
2863: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2864: Function VarIsArray( const A : Variant) : Boolean;
2865: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2866: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2867: Function VarArrayOf( const Values : array of Variant) : Variant
2868: Function VarArrayRef( const A : Variant) : Variant
2869: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2870: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2871: Function VarArrayDimCount( const A : Variant) : Integer
2872: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2873: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2874: Function VarArrayLock( const A : Variant) : ___Pointer
2875: Procedure VarArrayUnlock( const A : Variant)
```

```
2876: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2877: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2878: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2879: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2880: Function Unassigned : Variant
2881: Function Null : Variant
2882: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2883: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2884: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2885: function VectorDot(const V1,V2: TFloatPoint): Double;
2886: Function VectorLengthSqr( const V : TFloatPoint) : Double
2887: function VectorLengthSqr(const V: TFloatPoint): Double;
2888: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2889: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2890: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2891: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2892: Function Verify( AUserName : String) : String
2893: Function Versine( X : Float) : Float
2894: function VersionCheck: boolean;
2895: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2896: Function VersionLanguageName( const LangId : Word) : string
2897: Function VersionResourceAvailable( const FileName : string) : Boolean
2898: Function Visible : Boolean
2899: function VolumeID(DriveChar: Char): string
2900: Function WaitFor( const AString : string) : string
2901: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2902: Function WaitFor1 : TWaitResult;
2903: Function WaitForData( Timeout : Longint) : Boolean
2904: Function WebColorNameToColor( WebColorName : string) : TColor
2905: Function WebColorStrToColor( WebColor : string) : TColor
2906: Function WebColorToRGB( WebColor : Integer) : Integer
2907: Function wGet(aURL, afile: string): boolean;'
2908: Function wGet2(aURL, afile: string): boolean;'  //without file open
2909: Function WebGet(aURL, afile: string): boolean;'
2910: Function WebExists: boolean;   //alias to isinternet
2911: Function WeekOf( const AValue : TDateTime) : Word
2912: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2913: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2914: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2915: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2916: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2917: Function WeeksInAYear( const AYear : Word) : Word
2918: Function WeeksInYear( const AValue : TDateTime) : Word
2919: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2920: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2921: Function WideCat( const x, y : WideString) : WideString
2922: Function WideCompareStr( S1, S2 : WideString) : Integer
2923: function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
2924: Function WideCompareText( S1, S2 : WideString) : Integer
2925: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
2926: Function WideCopy( const src : WideString; index, count : Integer) : WideString
2927: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
2928: Function WideEqual( const x, y : WideString) : Boolean
2929: function WideFormat(const Format: WideString; const Args: array of const): WideString)
2930: Function WideGreater( const x, y : WideString) : Boolean
2931: Function WideLength( const src : WideString) : Integer
2932: Function WideLess( const x, y : WideString) : Boolean
2933: Function WideLowerCase( S : WideString) : WideString
2934: function WideLowerCase(const S: WideString): WideString)
2935: Function WidePos( const src, sub : WideString) : Integer
2936: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
2937: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
2938: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
2939: Function WideSameStr( S1, S2 : WideString) : Boolean
2940: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
2941: Function WideSameText( S1, S2 : WideString) : Boolean
2942: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
2943: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
2944: Function WideStringToUCS4String( const S : WideString) : UCS4String
2945: Function WideUpperCase( S : WideString) : WideString
2946: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
2947: function Win32Check(RetVal: boolean): boolean)
2948: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
2949: Function Win32RestoreFile( const FileName : string) : Boolean
2950: Function Win32Type : TIdWin32Type
2951: Function WinColor( const Color32 : TColor32) : TColor
2952: function winexec(FileName: pchar; showCmd: integer): integer;
2953: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
2954: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
2955: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
2956: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
2957: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
2958: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
2959: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
2960: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
2961: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
2962: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
2963: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
2964: Function WordToStr( const Value : Word) : String
```

```
2965: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean
2966: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
2967: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
2968: Function WorkArea : Integer
2969: Function WrapText( Line : string; MaxCol : Integer) : string;
2970: Function WrapText( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
2971: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
2972: function Write(Buffer:String;Count:LongInt):LongInt
2973: Function WriteClient( var Buffer, Count : Integer) : Integer
2974: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
2975: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
2976: Function WriteString( const AString : string) : Boolean
2977: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
2978: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
2979: Function wsprintf( Output : PChar; Format : PChar) : Integer
2980: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
2981: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
2982: Function XorDecode( const Key, Source : string) : string
2983: Function XorEncode( const Key, Source : string) : string
2984: Function XorString( const Key, Src : ShortString) : ShortString
2985: Function Yield : Bool
2986: Function YearOf( const AValue : TDateTime) : Word
2987: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
2988: Function YearSpan( const ANow, AThen : TDateTime) : Double
2989: Function Yesterday : TDateTime
2990: Function YesNoDialog(const ACaption, AMsg: string): boolean;
2991: Function( const Name : string; Proc : TUserFunction)
2992: Function using Special_Scholz from 3.8.5.0
2993: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
2994: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
2995: Function FloatToTime2Dec(value:Extended):Extended;
2996: Function MinToStd(value:Extended):Extended;
2997: Function MinToStdAsString(value:Extended):String;
2998: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
2999: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3000: Function Round2Dec (zahl:Extended):Extended;
3001: Function GetAngle(x,y:Extended):Double;
3002: Function AddAngle(a1,a2:Double):Double;
3003:
3004: ************************************************************************
3005: unit uPSI_StText;
3006: ************************************************************************
3007: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3008: Function TextFileSize( var F : TextFile) : LongInt
3009: Function TextPos( var F : TextFile) : LongInt
3010: Function TextFlush( var F : TextFile) : Boolean
3011:
3012: ************************************************************************
3013: from  JvVCLUtils;
3014: ************************************************************************
3015: { Windows resources (bitmaps and icons) VCL-oriented routines }
3016: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3017: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
      DstY:Int;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3018: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
      Bitmap: TBitmap; TransparentColor: TColor);
3019: function MakeBitmap(ResID: PChar): TBitmap;
3020: function MakeBitmapID(ResID: Word): TBitmap;
3021: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3022: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3023: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3024: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
3025:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3026: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3027: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3028: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3029: {$IFDEF WIN32}
3030: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3031:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3032: {$ENDIF}
3033: function MakeIcon(ResID: PChar): TIcon;
3034: function MakeIconID(ResID: Word): TIcon;
3035: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3036: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3037: {$IFDEF WIN32}
3038: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3039: {$ENDIF}
3040: { Service routines }
3041: procedure NotImplemented;
3042: procedure ResourceNotFound(ResID: PChar);
3043: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3044: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3045: function PaletteColor(Color: TColor): Longint;
3046: function WidthOf(R: TRect): Integer;
3047: function HeightOf(R: TRect): Integer;
3048: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3049: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3050: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3051: procedure Delay(MSecs: Longint);
```

```
3052: procedure CenterControl(Control: TControl);
3053: Function PaletteEntries( Palette : HPALETTE) : Integer
3054: Function WindowClassName( Wnd : HWND) : string
3055: Function ScreenWorkArea : TRect
3056: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3057: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3058: Procedure ActivateWindow( Wnd : HWND)
3059: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3060: Procedure CenterWindow( Wnd : HWND)
3061: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3062: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3063: Function DialogsToPixelsX( Dlgs : Word) : Word
3064: Function DialogsToPixelsY( Dlgs : Word) : Word
3065: Function PixelsToDialogsX( Pixs : Word) : Word
3066: Function PixelsToDialogsY( Pixs : Word) : Word
3067: {$IFDEF WIN32}
3068: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3069: function MakeVariant(const Values: array of Variant): Variant;
3070: {$ENDIF}
3071: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3072: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3073: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3074: {$IFDEF CBUILDER}
3075: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3076: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3077: {$ELSE}
3078: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3079: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3080: {$ENDIF CBUILDER}
3081: function IsForegroundTask: Boolean;
3082: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3083: function GetAveCharSize(Canvas: TCanvas): TPoint;
3084: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3085: procedure FreeUnusedOle;
3086: procedure Beep;
3087: function GetWindowsVersionJ: string;
3088: function LoadDLL(const LibName: string): THandle;
3089: function RegisterServer(const ModuleName: string): Boolean;
3090: {$IFNDEF WIN32}
3091: function IsLibrary: Boolean;
3092: {$ENDIF}
3093: { Gradient filling routine }
3094: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3095: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
     TFillDirection; Colors: Byte);
3096: { String routines }
3097: function GetEnvVar(const VarName: string): string;
3098: function AnsiUpperFirstChar(const S: string): string;
3099: function StringToPChar(var S: string): PChar;
3100: function StrPAlloc(const S: string): PChar;
3101: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3102: function DropT(const S: string): string;
3103: { Memory routines }
3104: function AllocMemo(Size: Longint): Pointer;
3105: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3106: procedure FreeMemo(var fpBlock: Pointer);
3107: function GetMemoSize(fpBlock: Pointer): Longint;
3108: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3109: {$IFNDEF COMPILER5_UP}
3110: procedure FreeAndNil(var Obj);
3111: {$ENDIF}
3112: // from PNGLoader
3113: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3114: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3115: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3116: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3117: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
     TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3118:   Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3119:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3120:  AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3121:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3122:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3123:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3124:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3125:  Procedure SetImeName( Name : TImeName)
3126:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3127:  Function Imm32GetContext( hWnd : HWND) : HIMC
3128:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3129:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
3130:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3131:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3132: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3133:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3134:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3135:  Function Imm32IsIME( hKl : longword) : Boolean
3136:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3137:  Procedure DragDone( Drop : Boolean)
3138:
```

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

```
3227: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3228: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3229: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
      HighEmpty:Double;Inclusive:Bool):string
3230: Function StrMaskSQL( const Value : string) : string
3231: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3232: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3233: Procedure _DBError( const Msg : string)
3234:  Const('TrueExpr','String '0=0
3235:  Const('sdfStandard16','String '''"''mm''/''dd''/''yyyy''"''
3236:  Const('sdfStandard32','String ''''''dd/mm/yyyy''''''
3237:  Const('sdfOracle','String '"TO_DATE(''"dd/mm/yyyy"'', ''DD/MM/YYYY'')"
3238:  Const('sdfInterbase','String '"CAST(''"mm"/"dd"/"yyyy"'' AS DATE)"
3239:  Const('sdfMSSQL','String '"CONVERT(datetime, ''"mm"/"dd"/"yyyy"'', 103)"
3240:  AddTypeS('Largeint', 'Longint
3241:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3242:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3243:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3244:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3245:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3246: (*----------------------------------------------------------------------------*)
3247: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3248: begin
3249:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3250:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3251:  Function JIniReadString( const FileName, Section, Line : string) : string
3252:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3253:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3254:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3255:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3256:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3257: end;
3258:
3259: (* === compile-time registration functions === *)
3260: (*----------------------------------------------------------------------------*)
3261: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3262: begin
3263:  'UnixTimeStart','LongInt'( 25569);
3264:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3265:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3266:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3267:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3268:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3269:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3270:  Function DayOfDate( const DateTime : TDateTime) : Integer
3271:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3272:  Function YearOfDate( const DateTime : TDateTime) : Integer
3273:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3274:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3275:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3276:  Function HourOfTime( const DateTime : TDateTime) : Integer
3277:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3278:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3279:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3280:  Function IsISOLongYear( const Year : Word) : Boolean;
3281:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3282:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3283:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3284:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3285:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3286:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3287:  Function JIsLeapYear( const Year : Integer) : Boolean;
3288:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3289:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3290:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3291:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3292:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3293:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3294:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3295:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3296:  Function HoursToMSecs( Hours : Integer) : Integer
3297:  Function MinutesToMSecs( Minutes : Integer) : Integer
3298:  Function SecondsToMSecs( Seconds : Integer) : Integer
3299:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3300:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3301:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3302:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3303:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3304:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3305:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3306:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3307:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3308:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3309:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3310:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3311:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3312:  Function DosDateTimeToStr( DateTime : Integer) : string
3313:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3314:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
```

```
3315:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3316:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3317:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3318:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3319:  Function FileTimeToStr( const FileTime : TFileTime) : string
3320:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3321:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3322:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3323:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3324:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3325:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3326:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3327:   TJclUnixTime32', 'Longword
3328:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3329:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3330:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3331:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3332:  Function JNullStamp : TTimeStamp
3333:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3334:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3335:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3336:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3337:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3338:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3339:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3340:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3341:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3342:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3343:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3344:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3345:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3346:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3347:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3348:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3349:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3350:   FindClass('TOBJECT'),'EJclDateTimeError
3351: end;
3352:
3353: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3354: begin
3355:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3356:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3357:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3358:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3359:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3360:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3361:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3362:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3363:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3364:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3365:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3366:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3367:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3368:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3369:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3370:  Function AbortShutDown : Boolean;
3371:  Function AbortShutDown1( const MachineName : string) : Boolean;
3372:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3373:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3374:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3375:   FindClass('TOBJECT'),'EJclCreateProcessError
3376:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3377:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
     Environment:PChar);
3378:   // with Add(EJclCreateProcessError) do
3379:  end;
3380:
3381:
3382: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3383: begin
3384:   //'AnsiSigns','Set').SetSet(['-', '+']);
3385:  'C1_UPPER','LongWord( $0001);
3386:  'C1_LOWER','LongWord( $0002);
3387:  'C1_DIGIT','LongWord').SetUInt( $0004);
3388:  'C1_SPACE','LongWord').SetUInt( $0008);
3389:  'C1_PUNCT','LongWord').SetUInt( $0010);
3390:  'C1_CNTRL','LongWord').SetUInt( $0020);
3391:  'C1_BLANK','LongWord').SetUInt( $0040);
3392:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3393:  'C1_ALPHA','LongWord').SetUInt( $0100);
3394:   AnsiChar', 'Char
3395:  Function StrIsAlpha( const S : AnsiString) : Boolean
3396:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3397:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3398:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3399:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3400:  Function StrIsDigit( const S : AnsiString) : Boolean
3401:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3402:  Function StrSame( const S1, S2 : AnsiString) : Boolean
```

```
3403:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3404:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3405:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3406:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3407:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3408:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3409:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3410:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3411:  Function StrEscapedToString( const S : AnsiString) : AnsiString
3412:  Function JStrLower( const S : AnsiString) : AnsiString
3413:  Procedure StrLowerInPlace( var S : AnsiString)
3414:  ///Procedure StrLowerBuff( S : PAnsiChar)
3415:  Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3416:  Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3417:  Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3418:  Function StrProper( const S : AnsiString) : AnsiString
3419:  //Procedure StrProperBuff( S : PAnsiChar)
3420:  Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3421:  Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3422:  Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3423:  Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3424:  Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3425:  Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3426:  Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3427:  Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3428:  Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3429:  Function StrReverse( const S : AnsiString) : AnsiString
3430:  Procedure StrReverseInPlace( var S : AnsiString)
3431:  Function StrSingleQuote( const S : AnsiString) : AnsiString
3432:  Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
3433:  Function StrStringToEscaped( const S : AnsiString) : AnsiString
3434:  Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3435:  Function StrToHex( const Source : AnsiString) : AnsiString
3436:  Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3437:  Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3438:  Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3439:  Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3440:  Function StrTrimQuotes( const S : AnsiString) : AnsiString
3441:  Function JStrUpper( const S : AnsiString) : AnsiString
3442:  Procedure StrUpperInPlace( var S : AnsiString)
3443:  //Procedure StrUpperBuff( S : PAnsiChar)
3444:  Function StrOemToAnsi( const S : AnsiString) : AnsiString
3445:  Function StrAnsiToOem( const S : AnsiString) : AnsiString
3446:  Procedure StrAddRef( var S : AnsiString)
3447:  Function StrAllocSize( const S : AnsiString) : Longint
3448:  Procedure StrDecRef( var S : AnsiString)
3449:  //Function StrLen( S : PAnsiChar) : Integer
3450:  Function StrLength( const S : AnsiString) : Longint
3451:  Function StrRefCount( const S : AnsiString) : Longint
3452:  Procedure StrResetLength( var S : AnsiString)
3453:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3454:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3455:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3456:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3457:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3458:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3459:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3460:  Function StrFillChar(const C: Char; Count: Integer): string)');
3461:  Function IntFillChar(const I: Integer; Count: Integer): string)');
3462:  Function ByteFillChar(const B: Byte; Count: Integer): string)');
3463:  Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;');
3464:  Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3465:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3466:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3467:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3468:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
3469:  Function StrIPos( const SubStr, S : AnsiString) : Integer
3470:  Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3471:  Function StrLastPos( const SubStr, S : AnsiString) : Integer
3472:  Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3473:  Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3474:  Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3475:  Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3476:  Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3477:  Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3478:  //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3479:  //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3480:  Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3481:  Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3482:  Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3483:  Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
3484:  Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3485:  Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3486:  Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3487:  Function CharIsAlpha( const C : AnsiChar) : Boolean
3488:  Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3489:  Function CharIsBlank( const C : AnsiChar) : Boolean
3490:  Function CharIsControl( const C : AnsiChar) : Boolean
3491:  Function CharIsDelete( const C : AnsiChar) : Boolean
```

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

```
3578: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)
3579: Function CurrentRecordDeleted( DataSet : TBDEDataSet) : Boolean
3580: Function IsFilterApplicable( DataSet : TDataSet) : Boolean
3581: Function IsBookmarkStable( DataSet : TBDEDataSet) : Boolean
3582: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3583: Procedure RestoreIndex( Table : TTable)
3584: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3585: Procedure PackTable( Table : TTable)
3586: Procedure ReindexTable( Table : TTable)
3587: Procedure BdeFlushBuffers
3588: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer
3589: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3590: Procedure DbNotSupported
3591: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
     AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3592: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
     AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3593: Procedure
     ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3594: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3595: ********************************************uPSI_JvDateUtil;
3596: function CurrentYear: Word;
3597: function IsLeapYear(AYear: Integer): Boolean;
3598: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3599: function FirstDayOfPrevMonth: TDateTime;
3600: function LastDayOfPrevMonth: TDateTime;
3601: function FirstDayOfNextMonth: TDateTime;
3602: function ExtractDay(ADate: TDateTime): Word;
3603: function ExtractMonth(ADate: TDateTime): Word;
3604: function ExtractYear(ADate: TDateTime): Word;
3605: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3606: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3607: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3608: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3609: function ValidDate(ADate: TDateTime): Boolean;
3610: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3611: function MonthsBetween(Date1, Date2: TDateTime): Double;
3612: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3613: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3614: function DaysBetween(Date1, Date2: TDateTime): Longint;
3615: { The same as previous but if Date2 < Date1 result = 0 }
3616: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3617: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3618: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3619: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3620: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3621: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3622: { String to date conversions }
3623: function GetDateOrder(const DateFormat: string): TDateOrder;
3624: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3625: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3626: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3627: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3628: function DefDateFormat(FourDigitYear: Boolean): string;
3629: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3630: ------------------------------------------------------------------------------
3631: ******************************** JvUtils;*****************************
3632: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3633: function GetWordOnPos(const S: string; const P: Integer): string;
3634: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3635: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3636: { SubStr returns substring from string, S, separated with Separator string}
3637: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3638: { SubStrEnd same to previous function but Index numerated from the end of string }
3639: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3640: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3641: function SubWord(P: PChar; var P2: PChar): string;
3642: { NumberByWord returns the text representation of
3643:   the number, N, in normal russian language. Was typed from Monitor magazine }
3644: function NumberByWord(const N: Longint): string;
3645: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3646:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3647: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3648: { GetXYByPos is same to previous function, but returns X position in line too}
3649: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3650: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3651: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3652: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3653: function ConcatSep(const S, S2, Separator: string): string;
3654: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3655: function ConcatLeftSep(const S, S2, Separator: string): string;
3656: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3657: function MinimizeString(const S: string; const MaxLen: Integer): string;
3658: { Next 4 function for russian chars transliterating.
3659:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3660: procedure Dos2Win(var S: string);
3661: procedure Win2Dos(var S: string);
3662: function Dos2WinRes(const S: string): string;
3663: function Win2DosRes(const S: string): string;
```

```
3664: function Win2Koi(const S: string): string;
3665: { Spaces returns string consists on N space chars }
3666: function Spaces(const N: Integer): string;
3667: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3668: function AddSpaces(const S: string; const N: Integer): string;
3669: { function LastDate for russian users only } { returns date relative to current date: '' }
3670: function LastDate(const Dat: TDateTime): string;
3671: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3672: function CurrencyToStr(const Cur: currency): string;
3673: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3674: function Cmp(const S1, S2: string): Boolean;
3675: { StringCat add S2 string to S1 and returns this string }
3676: function StringCat(var S1: string; S2: string): string;
3677: { HasChar returns True, if Char, Ch, contains in string, S }
3678: function HasChar(const Ch: Char; const S: string): Boolean;
3679: function HasAnyChar(const Chars: string; const S: string): Boolean;
3680: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3681: function CountOfChar(const Ch: Char; const S: string): Integer;
3682: function DefStr(const S: string; Default: string): string;
3683: {**** files routines}
3684: { GetWinDir returns Windows folder name }
3685: function GetWinDir: TFileName;
3686: function GetSysDir: String;
3687: { GetTempDir returns Windows temporary folder name }
3688: function GetTempDir: string;
3689: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3690: function GenTempFileName(FileName: string): string;
3691: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3692: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3693: { ClearDir clears folder Dir }
3694: function ClearDir(const Dir: string): Boolean;
3695: { DeleteDir clears and than delete folder Dir }
3696: function DeleteDir(const Dir: string): Boolean;
3697: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3698: function FileEquMask(FileName, Mask: TFileName): Boolean;
3699: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3700:   Masks must be separated with comma (';') }
3701: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3702: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3703: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3704: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3705: { FileGetInfo fills SearchRec record for specified file attributes}
3706: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3707: { HasSubFolder returns True, if folder APath contains other folders }
3708: function HasSubFolder(APath: TFileName): Boolean;
3709: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3710: function IsEmptyFolder(APath: TFileName): Boolean;
3711: { AddSlash add slash Char to Dir parameter, if needed }
3712: procedure AddSlash(var Dir: TFileName);
3713: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3714: function AddSlash2(const Dir: TFileName): string;
3715: { AddPath returns FileName with Path, if FileName not contain any path }
3716: function AddPath(const FileName, Path: TFileName): TFileName;
3717: function AddPaths(const PathList, Path: string): string;
3718: function ParentPath(const Path: TFileName): TFileName;
3719: function FindInPath(const FileName, PathList: string): TFileName;
3720: function FindInPaths(const fileName,paths: String): String;
3721: {$IFNDEF BCB1}
3722: { BrowseForFolder displays Browse For Folder dialog }
3723: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3724: {$ENDIF BCB1}
3725: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3726: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3727: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3728: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3729:
3730: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3731: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3732: { HasParam returns True, if program running with specified parameter, Param }
3733: function HasParam(const Param: string): Boolean;
3734: function HasSwitch(const Param: string): Boolean;
3735: function Switch(const Param: string): string;
3736: { ExePath returns ExtractFilePath(ParamStr(0)) }
3737: function ExePath: TFileName;
3738: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3739: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3740: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3741: {**** Graphic routines }
3742: { TTFontSelected returns True, if True Type font is selected in specified device context }
3743: function TTFontSelected(const DC: HDC): Boolean;
3744: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3745: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3746: {**** Windows routines }
3747: { SetWindowTop put window to top without recreating window }
3748: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3749: {**** other routines }
3750: { KeyPressed returns True, if Key VK is now pressed }
```

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

```
{ FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
function FindFormByClass(FormClass: TFormClass): TForm;
function FindFormByClassName(FormClassName: string): TForm;
{ FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
  having Tag property value, equaled to Tag parameter }
function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
{ ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
{ RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
function RBTag(Parent: TWinControl): Integer;
{ AppMinimized returns True, if Application is minimized }
function AppMinimized: Boolean;
{ MessageBox is Application.MessageBox with string (not PChar) parameters.
  if Caption parameter = '', it replaced with Application.Title }
function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
  Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
  Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
{ Delay stop program execution to MSec msec }
procedure Delay(MSec: Longword);
procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
procedure EnableControls(Control: TWinControl; const Enable: Boolean);
procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
function PanelBorder(Panel: TCustomPanel): Integer;
function Pixels(Control: TControl; APixels: Integer): Integer;
procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
procedure Error(const Msg: string);
procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
  {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean): string;
function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean): Integer;
function ItemHtPlain(const Text: string): string;
{ ClearList - clears list of TObject }
procedure ClearList(List: TList);
procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
{ RTTI support }
function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
function GetPropStr(Obj: TObject; const PropName: string): string;
function GetPropOrd(Obj: TObject; const PropName: string): Integer;
function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
procedure PrepareIniSection(SS: TStrings);
{ following functions are not documented because they are don't work properly, so don't use them }
{$IFDEF COMPILER2}
function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
{$ENDIF}

procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
begin
 Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
 Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
 Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
 Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
 Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
 Procedure BoxSetItem( List : TWinControl; Index : Integer)
 Function BoxGetFirstSelection( List : TWinControl) : Integer
 Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
end;

procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
begin
 Const('MaxInitStrNum','LongInt'( 9);
Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
 : array of AnsiString; MaxSplit : Integer) : Integer
Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
string; MaxSplit : Integer) : Integer
Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
Function JvAnsiStrStrip( S : AnsiString) : AnsiString
Function JvStrStrip( S : string) : string
Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
Function StrEatWhiteSpace( const S : string) : string
Function HexToAscii( const S : AnsiString) : AnsiString
Function AsciiToHex( const S : AnsiString) : AnsiString
Function StripQuotes( const S1 : AnsiString) : AnsiString
Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
Function HexPCharToInt( S1 : PAnsiChar) : Integer
Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
```

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

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

```
4099: function DeleteFiles(const FileMask: string): Boolean;
4100: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4101: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4102: function NormalDir(const DirName: string): string;
4103: function RemoveBackSlash(const DirName: string): string;
4104: function ValidFileName(const FileName: string): Boolean;
4105: function DirExists(Name: string): Boolean;
4106: procedure ForceDirectories(Dir: string);
4107: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4108:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4109: {$IFDEF COMPILER4_UP}
4110: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4111: {$ENDIF}
4112: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4113: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4114: {$IFDEF COMPILER4_UP}
4115: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4116: {$ENDIF}
4117: function GetTempDir: string;
4118: function GetWindowsDir: string;
4119: function GetSystemDir: string;
4120: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4121: {$IFDEF WIN32}
4122: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4123: function ShortToLongFileName(const ShortName: string): string;
4124: function ShortToLongPath(const ShortName: string): string;
4125: function LongToShortFileName(const LongName: string): string;
4126: function LongToShortPath(const LongName: string): string;
4127: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4128: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4129: {$ENDIF WIN32}
4130: {$IFNDEF COMPILER3_UP}
4131: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4132: {$ENDIF}
4133: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4134: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4135: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4136: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4137:
4138: //**************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4139:  Procedure VariantClear( var V : Variant)
4140:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4141:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4142:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4143:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4144:  Procedure VariantSub( const src : Variant; var dst : Variant)
4145:  Procedure VariantMul( const src : Variant; var dst : Variant)
4146:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4147:  Procedure VariantMod( const src : Variant; var dst : Variant)
4148:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4149:  Procedure VariantOr( const src : Variant; var dst : Variant)
4150:  Procedure VariantXor( const src : Variant; var dst : Variant)
4151:  Procedure VariantShl( const src : Variant; var dst : Variant)
4152:  Procedure VariantShr( const src : Variant; var dst : Variant)
4153:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4154:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4155:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4156:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4157:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4158:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4159:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4160:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4161:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4162:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4163:  Function VariantNot( const V1 : Variant) : Variant
4164:  Function VariantNeg( const V1 : Variant) : Variant
4165:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4166:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4167:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4168:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4169:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4170:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4171:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4172:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4173:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4174:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4175: end;
4176:
4177: ************************************unit uPSI_JvgUtils;************************************
4178: function IsEven(I: Integer): Boolean;
4179: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4180: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4181: procedure SwapInt(var I1, I2: Integer);
4182: function Spaces(Count: Integer): string;
4183: function DupStr(const Str: string; Count: Integer): string;
4184: function DupChar(C: Char; Count: Integer): string;
4185: procedure Msg(const AMsg: string);
4186: function RectW(R: TRect): Integer;
4187: function RectH(R: TRect): Integer;
```

```
4188: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4189: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4190: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4191: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4192:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4193: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4194: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4195:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
     TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4196: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4197: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4198:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4199: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4200: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4201: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4202:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4203:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4204:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4205: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4206:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4207:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4208:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4209: procedure BringParentWindowToTop(Wnd: TWinControl);
4210: function GetParentForm(Control: TControl): TForm;
4211: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4212: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4213: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4214: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4215: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4216: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4217: function CalcMathString(AExpression: string): Single;
4218: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4219: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4220: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4221: procedure TypeStringOnKeyboard(const S: string);
4222: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4223: procedure DrawTextExtAligned(Canvas:TCanvas;const
     Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4224: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4225: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4226: function ComponentToString(Component: TComponent): string;
4227: procedure StringToComponent(Component: TComponent; const Value: string);
4228: function PlayWaveResource(const ResName: string): Boolean;
4229: function UserName: string;
4230: function ComputerName: string;
4231: function CreateIniFileName: string;
4232: function ExpandString(const Str: string; Len: Integer): string;
4233: function Transliterate(const Str: string; RusToLat: Boolean): string;
4234: function IsSmallFonts: Boolean;
4235: function SystemColorDepth: Integer;
4236: function GetFileTypeJ(const FileName: string): TglFileType;
4237: Function GetFileType( hFile : THandle) : DWORD');
4238: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4239: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4240:
4241: { ************************************Utility routines of unit classes}
4242: function LineStart(Buffer, BufPos: PChar): PChar
4243: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4244:   'Strings: TStrings): Integer
4245:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4246:  Procedure RegisterClass( AClass : TPersistentClass)
4247:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4248:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4249:  Procedure UnRegisterClass( AClass : TPersistentClass)
4250:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4251:  Procedure UnRegisterModuleClasses( Module : HMODULE)
4252:  Function FindGlobalComponent( const Name : string) : TComponent
4253:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4254:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4255:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4256:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4257:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4258:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4259:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4260:  Procedure GlobalFixupReferences
4261:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4262:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4263:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4264:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4265:  Procedure RemoveFixups( Instance : TPersistent)
4266:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4267:  Procedure BeginGlobalLoading
4268:  Procedure NotifyGlobalLoading
4269:  Procedure EndGlobalLoading
4270:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4271:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4272:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4273:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4274:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
```

```
4275: // Function AllocateHWnd( Method : TWndMethod) : HWND
4276:  Procedure DeallocateHWnd( Wnd : HWND)
4277:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4278: ************************************unit uPSI_SqlTimSt and DB;*********************************
4279: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4280: Function VarSQLTimeStampCreate3: Variant;
4281: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4282: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4283: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4284: Function VarSQLTimeStamp : TVarType
4285: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4286: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4287: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4288: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4289: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4290: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4291: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4292: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4293: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4294: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4295: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4296: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4297: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4298:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4299: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4300: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4301: Procedure DisposeMem( var Buffer, Size : Integer)
4302: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4303: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4304: Function VarTypeToDataType( VarType : Integer) : TFieldType
4305: ******************************************unit JvStrings;*************************************
4306: {template functions}
4307: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4308: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4309: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4310: function RemoveMasterBlocks(const SourceStr: string): string;
4311: function RemoveFields(const SourceStr: string): string;
4312: {http functions}
4313: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4314: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4315: {set functions}
4316: procedure SplitSet(AText: string; AList: TStringList);
4317: function JoinSet(AList: TStringList): string;
4318: function FirstOfSet(const AText: string): string;
4319: function LastOfSet(const AText: string): string;
4320: function CountOfSet(const AText: string): Integer;
4321: function SetRotateRight(const AText: string): string;
4322: function SetRotateLeft(const AText: string): string;
4323: function SetPick(const AText: string; AIndex: Integer): string;
4324: function SetSort(const AText: string): string;
4325: function SetUnion(const Set1, Set2: string): string;
4326: function SetIntersect(const Set1, Set2: string): string;
4327: function SetExclude(const Set1, Set2: string): string;
4328: {replace any <,> etc by &lt; &gt;}
4329: function XMLSafe(const AText: string): string;
4330: {simple hash, Result can be used in Encrypt}
4331: function Hash(const AText: string): Integer;
4332: { Base64 encode and decode a string }
4333: function B64Encode(const S: AnsiString): AnsiString;
4334: function B64Decode(const S: AnsiString): AnsiString;
4335: {Basic encryption from a Borland Example}
4336: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4337: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4338: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4339: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4340: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4341: procedure CSVToTags(Src, Dst: TStringList);
4342: // converts a csv list to a tagged string list
4343: procedure TagsToCSV(Src, Dst: TStringList);
4344: // converts a tagged string list to a csv list
4345: // only fieldnames from the first record are scanned ib the other records
4346: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4347: {selects akey=avalue from Src and returns recordset in Dst}
4348: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4349: {filters Src for akey=avalue}
4350: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4351: {orders a tagged Src list by akey}
4352: function PosStr(const FindString, SourceString: string;
4353:   StartPos: Integer = 1): Integer;
4354: { PosStr searches the first occurrence of a substring FindString in a string
4355:   given by SourceString with case sensitivity (upper and lower case characters
4356:   are differed). This function returns the index value of the first character
4357:   of a specified substring from which it occurs in a given string starting with
4358:   StartPos character index. If a specified substring is not found Q_PosStr
4359:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4360: function PosStrLast(const FindString, SourceString: string): Integer;
4361: {finds the last occurance}
4362: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4363: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
```

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

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

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

```
4617: { ExePath returns ExtractFilePath(ParamStr(0)) }
4618: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4619: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4620: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4621: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4622: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4623: {**** Graphic routines }
4624: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4625: function IsTTFontSelected(const DC: HDC): Boolean;
4626: function KeyPressed(VK: Integer): Boolean;
4627: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4628: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4629: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4630: {**** Color routines }
4631: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4632: function RGBToBGR(Value: Cardinal): Cardinal;
4633: //function ColorToPrettyName(Value: TColor): string;
4634: //function PrettyNameToColor(const Value: string): TColor;
4635: {**** other routines }
4636: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4637: function IntPower(Base, Exponent: Integer): Integer;
4638: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4639: function StrToBool(const S: string): Boolean;
4640: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4641: function VarToInt(V: Variant): Integer;
4642: function VarToFloat(V: Variant): Double;
4643: { following functions are not documented because they not work properly sometimes, so do not use them }
4644: // (rom) ReplaceStrings1, GetSubStr removed
4645: function GetLongFileName(const FileName: string): string;
4646: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4647: function GetParameter: string;
4648: function GetComputerID: string;
4649: function GetComputerName: string;
4650: {**** string routines }
4651: { ReplaceAllStrings searches for all substrings, Words,
4652:   in a string, S, and replaces them with Frases with the same Index. }
4653: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4654: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4655:   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 }
4656: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4657: { CountOfLines calculates the lines count in a string, S,
4658:   each line must be separated from another with CrLf sequence }
4659: function CountOfLines(const S: string): Integer;
4660: { DeleteLines deletes all lines from strings which in the words,  words.
4661:   The word of will be deleted from strings. }
4662: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4663: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4664:   Lines contained only spaces also deletes. }
4665: procedure DeleteEmptyLines(Ss: TStrings);
4666: { SQLAddWhere addes or modifies existing where-statement, where,
4667:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4668:   it must be started on the begining of any line }
4669: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4670: {**** files routines - }
4671: {$IFDEF MSWINDOWS}
4672: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4673:   Resource can be compressed using MS Compress program}
4674: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4675: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
      Boolean;
4676: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4677: {$ENDIF MSWINDOWS}
4678: { IniReadSection read section, Section, from ini-file,
4679:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4680:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4681: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4682: { LoadTextFile load text file, FileName, into string }
4683: function LoadTextFile(const FileName: TFileName): string;
4684: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4685: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4686: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4687: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4688: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4689: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4690: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4691: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4692: { RATextCalcHeight calculate needed height for
4693:   correct output, using RATextOut or RATextOutEx functions }
4694: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4695: { Cinema draws some visual effect }
4696: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4697: { Roughed fills rect with special 3D pattern }
4698: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4699: { 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 }
4700: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4701: { TextWidth calculate text with for writing using standard desktop font }
4702: function TextWidth(const AStr: string): Integer;
```

```
4703: { TextHeight calculate text height for writing using standard desktop font }
4704: function TextHeight(const AStr: string): Integer;
4705: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4706: procedure Error(const Msg: string);
4707: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4708:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4709: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4710: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4711:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4712: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4713:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4714: function ItemHtPlain(const Text: string): string;
4715: { ClearList - clears list of TObject }
4716: procedure ClearList(List: TList);
4717: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4718: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4719: { RTTI support }
4720: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4721: function GetPropStr(Obj: TObject; const PropName: string): string;
4722: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4723: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4724: procedure PrepareIniSection(Ss: TStrings);
4725: { following functions are not documented because they are don't work properly, so don't use them }
4726: // (rom) from JvBandWindows to make it obsolete
4727: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4728: // (rom) from JvBandUtils to make it obsolete
4729: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4730: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4731: function CreateIconFromClipboard: TIcon;
4732: { begin JvIconClipboardUtils } { Icon clipboard routines }
4733: function CF_ICON: Word;
4734: procedure AssignClipboardIcon(Icon: TIcon);
4735: { Real-size icons support routines (32-bit only) }
4736: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4737: function CreateRealSizeIcon(Icon: TIcon): HICON;
4738: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4739: {end JvIconClipboardUtils }
4740: function CreateScreenCompatibleDC: HDC;
4741: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF}
4742: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4743: { begin JvRLE } // (rom) changed API for inclusion in JCL
4744: procedure RleCompressTo(InStream, OutStream: TStream);
4745: procedure RleDecompressTo(InStream, OutStream: TStream);
4746: procedure RleCompress(Stream: TStream);
4747: procedure RleDecompress(Stream: TStream);
4748: { end JvRLE } { begin JvDateUtil }
4749: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4750: function IsLeapYear(AYear: Integer): Boolean;
4751: function DaysInAMonth(const AYear, AMonth: Word): Word;
4752: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4753: function FirstDayOfPrevMonth: TDateTime;
4754: function LastDayOfPrevMonth: TDateTime;
4755: function FirstDayOfNextMonth: TDateTime;
4756: function ExtractDay(ADate: TDateTime): Word;
4757: function ExtractMonth(ADate: TDateTime): Word;
4758: function ExtractYear(ADate: TDateTime): Word;
4759: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4760: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4761: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4762: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4763: function ValidDate(ADate: TDateTime): Boolean;
4764: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4765: function MonthsBetween(Date1, Date2: TDateTime): Double;
4766: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4767: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4768: function DaysBetween(Date1, Date2: TDateTime): Longint;
4769: { The same as previous but if Date2 < Date1 result = 0 }
4770: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4771: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4772: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4773: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4774: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4775: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4776: { String to date conversions }
4777: function GetDateOrder(const DateFormat: string): TDateOrder;
4778: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4779: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4780: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4781: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4782: //function DefDateFormat(AFourDigitYear: Boolean): string;
4783: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4784: function FormatLongDate(Value: TDateTime): string;
4785: function FormatLongDateTime(Value: TDateTime): string;
4786: { end JvDateUtil }
4787: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4788: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4789: { begin JvStrUtils } { ** Common string handling routines ** }
4790: {$IFDEF UNIX}
```

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

```
4880: function XorString(const Key, Src: ShortString): ShortString;
4881: function XorEncode(const Key, Source: string): string;
4882: function XorDecode(const Key, Source: string): string;
4883: { ** Command line routines ** }
4884: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4885: { ** Numeric string handling routines ** }
4886: function Numb2USA(const S: string): string;
4887: { Numb2USA converts numeric string S to USA-format. }
4888: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4889: { Dec2Hex converts the given value to a hexadecimal string representation
4890:   with the minimum number of digits (A) specified. }
4891: function Hex2Dec(const S: string): Longint;
4892: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4893: function Dec2Numb(N: Int64; A, B: Byte): string;
4894: { Dec2Numb converts the given value to a string representation with the
4895:   base equal to B and with the minimum number of digits (A) specified. }
4896: function Numb2Dec(S: string; B: Byte): Int64;
4897: { Numb2Dec converts the given B-based numeric string to the corresponding
4898:   integer value. }
4899: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4900: { IntToBin converts the given value to a binary string representation
4901:   with the minimum number of digits specified. }
4902: function IntToRoman(Value: Longint): string;
4903: { IntToRoman converts the given value to a roman numeric string representation. }
4904: function RomanToInt(const S: string): Longint;
4905: { RomanToInt converts the given string to an integer value. If the string
4906:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4907: function FindNotBlankCharPos(const S: string): Integer;
4908: function FindNotBlankCharPosW(const S: WideString): Integer;
4909: function AnsiChangeCase(const S: string): string;
4910: function WideChangeCase(const S: string): string;
4911: function StartsText(const SubStr, S: string): Boolean;
4912: function EndsText(const SubStr, S: string): Boolean;
4913: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4914: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4915: {end JvStrUtils}
4916: {$IFDEF UNIX}
4917: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4918: {$ENDIF UNIX}
4919: { begin JvFileUtil }
4920: function FileDateTime(const FileName: string): TDateTime;
4921: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4922: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4923: function NormalDir(const DirName: string): string;
4924: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4925: function ValidFileName(const FileName: string): Boolean;
4926: {$IFDEF MSWINDOWS}
4927: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4928: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4929: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4930: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4931: {$ENDIF MSWINDOWS}
4932: function GetWindowsDir: string;
4933: function GetSystemDir: string;
4934: function ShortToLongFileName(const ShortName: string): string;
4935: function LongToShortFileName(const LongName: string): string;
4936: function ShortToLongPath(const ShortName: string): string;
4937: function LongToShortPath(const LongName: string): string;
4938: {$IFDEF MSWINDOWS}
4939: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4940: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4941: {$ENDIF MSWINDOWS}
4942: { end JvFileUtil }
4943: // Works like PtInRect but includes all edges in comparision
4944: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4945: // Works like PtInRect but excludes all edges from comparision
4946: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4947: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4948: function IsFourDigitYear: Boolean;
4949: { moved from JvJVCLUTils }
4950: //Open an object with the shell (url or something like that)
4951: function OpenObject(const Value: string): Boolean; overload;
4952: function OpenObject(Value: PChar): Boolean; overload;
4953: {$IFDEF MSWINDOWS}
4954: //Raise the last Exception
4955: procedure RaiseLastWin32; overload;
4956: procedure RaiseLastWin32(const Text: string); overload;
4957: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
4958: function GetFileVersion(const AFileName: string): Cardinal;
4959: {$EXTERNALSYM GetFileVersion}
4960: //Get version of Shell.dll
4961: function GetShellVersion: Cardinal;
4962: {$EXTERNALSYM GetShellVersion}
4963: // CD functions  on HW
4964: procedure OpenCdDrive;
4965: procedure CloseCdDrive;
4966: // returns True if Drive is accessible
```

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

```
5053: { String routines }
5054: function GetEnvVar(const VarName: string): string;
5055: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5056: function StringToPChar(var S: string): PChar;
5057: function StrPAlloc(const S: string): PChar;
5058: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5059: function DropT(const S: string): string;
5060: { Memory routines }
5061: function AllocMemo(Size: Longint): Pointer;
5062: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5063: procedure FreeMemo(var fpBlock: Pointer);
5064: function GetMemoSize(fpBlock: Pointer): Longint;
5065: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5066: { Manipulate huge pointers routines }
5067: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5068: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5069: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5070: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5071: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5072: function WindowClassName(Wnd: THandle): string;
5073: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5074: procedure ActivateWindow(Wnd: THandle);
5075: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5076: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5077: { SetWindowTop put window to top without recreating window }
5078: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5079: procedure CenterWindow(Wnd: THandle);
5080: function MakeVariant(const Values: array of Variant): Variant;
5081: { Convert dialog units to pixels and backwards }
5082: {$IFDEF MSWINDOWS}
5083: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5084: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5085: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5086: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5087: {$ENDIF MSWINDOWS}
5088: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5089: {$IFDEF BCB}
5090: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5091: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5092: {$ELSE}
5093: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5094: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5095: {$ENDIF BCB}
5096: {$IFDEF MSWINDOWS}
5097: { BrowseForFolderNative displays Browse For Folder dialog }
5098: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5099: {$ENDIF MSWINDOWS}
5100: procedure AntiAlias(Clip: TBitmap);
5101: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5102: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5103:   ABitmap: TBitmap; const SourceRect: TRect);
5104: function IsTrueType(const FontName: string): Boolean;
5105: // Removes all non-numeric characters from AValue and returns the resulting string
5106: function TextToValText(const AValue: string): string;
5107: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5108: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5109: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5110: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5111: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5112:
5113: ******************************************************unit uPSI_JvTFUtils;
5114:  Function JExtractYear( ADate : TDateTime) : Word
5115:  Function JExtractMonth( ADate : TDateTime) : Word
5116:  Function JExtractDay( ADate : TDateTime) : Word
5117:  Function ExtractHours( ATime : TDateTime) : Word
5118:  Function ExtractMins( ATime : TDateTime) : Word
5119:  Function ExtractSecs( ATime : TDateTime) : Word
5120:  Function ExtractMSecs( ATime : TDateTime) : Word
5121:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5122:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5123:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5124:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5125:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5126:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5127:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5128:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5129:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5130:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5131:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5132:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5133:  Procedure EnsureMonth( Month : Word)
5134:  Procedure EnsureDOW( DOW : Word)
5135:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5136:  Function Lesser( N1, N2 : Integer) : Integer
5137:  Function Greater( N1, N2 : Integer) : Integer
5138:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5139:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5140:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
```

```
5141:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5142:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5143:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5144:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
       AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5145:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
       HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5146:  Function JRectWidth( ARect : TRect) : Integer
5147:  Function JRectHeight( ARect : TRect) : Integer
5148:  Function JEmptyRect : TRect
5149:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5150:
5151: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5152: begin
5153:  Procedure HideTaskBarButton( hWindow : HWND)
5154:  Function msLoadStr( ID : Integer) : String
5155:  Function msFormat( fmt : String; params : array of const) : String
5156:  Function msFileExists( const FileName : String) : Boolean
5157:  Function msIntToStr( Int : Int64) : String
5158:  Function msStrPas( const Str : PChar) : String
5159:  Function msRenameFile( const OldName, NewName : String) : Boolean
5160:  Function CutFileName( s : String) : String
5161:  Function GetVersionInfo( var VersionString : String) : DWORD
5162:  Function FormatTime( t : Cardinal) : String
5163:  Function msCreateDir( const Dir : string) : Boolean
5164:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5165:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5166:  Function msStrLen( Str : PChar) : Integer
5167:  Function msDirectoryExists( const Directory : String) : Boolean
5168:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5169:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5170:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5171:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5172:  Function GetTextFromFile( Filename : String) : string
5173:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5174:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5175:  Function msStrToInt( s : String) : Integer
5176:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
5177: end;
5178:
5179: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5180: begin
5181:   //TDynFloatArray', 'array of Extended
5182:   TDynLWordArray', 'array of LongWord
5183:   TDynLIntArray', 'array of LongInt
5184:   TDynFloatMatrix', 'array of TDynFloatArray
5185:   TDynLWordMatrix', 'array of TDynLWordArray
5186:   TDynLIntMatrix', 'array of TDynLIntArray
5187:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5188:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5189:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5190:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5191:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5192:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5193:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5194:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5195:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5196:  Function MNorm( const X : TDynFloatArray) : Extended
5197:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5198:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5199:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5200:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5201:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5202:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5203:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5204:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5205:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5206:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5207:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5208:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5209:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5210: end;
5211:
5212: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5213: begin
5214:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5215:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5216:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5217:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5218:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5219:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5220:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5221:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5222:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5223:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5224:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5225:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5226:  'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5227:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
```

```
5228:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5229:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5230:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5231:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5232:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5233:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5234:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5235:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5236:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5237:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5238:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5239:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5240:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5241:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5242:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5243:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
5244:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5245:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5246:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5247:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5248:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5249:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5250:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5251:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5252:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5253:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5254:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5255:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5256:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5257:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5258:  'ESBPiOn4','Extended').setExtended( 0.78539816339744830 96);
5259:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5260:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5261:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5262:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5263:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5264:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5265:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5266:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5267:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5268:   //LongWord', 'Cardinal
5269:   TBitList', 'Word
5270:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5271:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5272:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5273:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5274:  Function SameFloat( const X1, X2 : Extended) : Boolean
5275:  Function FloatIsZero( const X : Extended) : Boolean
5276:  Function FloatIsPositive( const X : Extended) : Boolean
5277:  Function FloatIsNegative( const X : Extended) : Boolean
5278:  Procedure IncLim( var B : Byte; const Limit : Byte)
5279:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5280:  Procedure IncLimW( var B : Word; const Limit : Word)
5281:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5282:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5283:  Procedure DecLim( var B : Byte; const Limit : Byte)
5284:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5285:  Procedure DecLimW( var B : Word; const Limit : Word)
5286:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5287:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5288:  Function MaxB( const B1, B2 : Byte) : Byte
5289:  Function MinB( const B1, B2 : Byte) : Byte
5290:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5291:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5292:  Function MaxW( const B1, B2 : Word) : Word
5293:  Function MinW( const B1, B2 : Word) : Word
5294:  Function esbMaxI( const B1, B2 : Integer) : Integer
5295:  Function esbMinI( const B1, B2 : Integer) : Integer
5296:  Function MaxL( const B1, B2 : LongInt) : LongInt
5297:  Function MinL( const B1, B2 : LongInt) : LongInt
5298:  Procedure SwapB( var B1, B2 : Byte)
5299:  Procedure SwapSI( var B1, B2 : ShortInt)
5300:  Procedure SwapW( var B1, B2 : Word)
5301:  Procedure SwapI( var B1, B2 : SmallInt)
5302:  Procedure SwapL( var B1, B2 : LongInt)
5303:  Procedure SwapI32( var B1, B2 : Integer)
5304:  Procedure SwapC( var B1, B2 : LongWord)
5305:  Procedure SwapInt64( var X, Y : Int64)
5306:  Function esbSign( const B : LongInt) : ShortInt
5307:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5308:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5309:  Function Max3Word( const X1, X2, X3 : Word) : Word
5310:  Function Min3Word( const X1, X2, X3 : Word) : Word
5311:  Function MaxBArray( const B : array of Byte) : Byte
5312:  Function MaxWArray( const B : array of Word) : Word
5313:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5314:  Function MaxIArray( const B : array of Integer) : Integer
5315:  Function MaxLArray( const B : array of LongInt) : LongInt
5316:  Function MinBArray( const B : array of Byte) : Byte
```

```
5317:  Function MinWArray( const B : array of Word) : Word
5318:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5319:  Function MinIArray( const B : array of Integer) : Integer
5320:  Function MinLArray( const B : array of LongInt) : LongInt
5321:  Function SumBArray( const B : array of Byte) : Byte
5322:  Function SumBArray2( const B : array of Byte) : Word
5323:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5324:  Function SumSIArray2( const B : array of ShortInt) : Integer
5325:  Function SumWArray( const B : array of Word) : Word
5326:  Function SumWArray2( const B : array of Word) : LongInt
5327:  Function SumIArray( const B : array of Integer) : Integer
5328:  Function SumLArray( const B : array of LongInt) : LongInt
5329:  Function SumLWArray( const B : array of LongWord) : LongWord
5330:  Function ESBDigits( const X : LongWord) : Byte
5331:  Function BitsHighest( const X : LongWord) : Integer
5332:  Function ESBBitsNeeded( const X : LongWord) : Integer
5333:  Function esbGCD( const X, Y : LongWord) : LongWord
5334:  Function esbLCM( const X, Y : LongInt) : Int64
5335:  //Function esbLCM( const X, Y : LongInt) : LongInt
5336:  Function RelativePrime( const X, Y : LongWord) : Boolean
5337:  Function Get87ControlWord : TBitList
5338:  Procedure Set87ControlWord( const CWord : TBitList)
5339:  Procedure SwapExt( var X, Y : Extended)
5340:  Procedure SwapDbl( var X, Y : Double)
5341:  Procedure SwapSing( var X, Y : Single)
5342:  Function esbSgn( const X : Extended) : ShortInt
5343:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5344:  Function ExtMod( const X, Y : Extended) : Extended
5345:  Function ExtRem( const X, Y : Extended) : Extended
5346:  Function CompMOD( const X, Y : Comp) : Comp
5347:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5348:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5349:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5350:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5351:  Function MaxExt( const X, Y : Extended) : Extended
5352:  Function MinExt( const X, Y : Extended) : Extended
5353:  Function MaxEArray( const B : array of Extended) : Extended
5354:  Function MinEArray( const B : array of Extended) : Extended
5355:  Function MaxSArray( const B : array of Single) : Single
5356:  Function MinSArray( const B : array of Single) : Single
5357:  Function MaxCArray( const B : array of Comp) : Comp
5358:  Function MinCArray( const B : array of Comp) : Comp
5359:  Function SumSArray( const B : array of Single) : Single
5360:  Function SumEArray( const B : array of Extended) : Extended
5361:  Function SumSqEArray( const B : array of Extended) : Extended
5362:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5363:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5364:  Function SumCArray( const B : array of Comp) : Comp
5365:  Function FactorialX( A : LongWord) : Extended
5366:  Function PermutationX( N, R : LongWord) : Extended
5367:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5368:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5369:  Function esbGeometricMean( const X : array of Extended) : Extended
5370:  Function esbHarmonicMean( const X : array of Extended) : Extended
5371:  Function ESBMean( const X : array of Extended) : Extended
5372:  Function esbSampleVariance( const X : array of Extended) : Extended
5373:  Function esbPopulationVariance( const X : array of Extended) : Extended
5374:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5375:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5376:  Function GetMedian( const SortedX : array of Extended) : Extended
5377:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5378:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5379:  Function ESBMagnitude( const X : Extended) : Integer
5380:  Function ESBTan( Angle : Extended) : Extended
5381:  Function ESBCot( Angle : Extended) : Extended
5382:  Function ESBCosec( const Angle : Extended) : Extended
5383:  Function ESBSec( const Angle : Extended) : Extended
5384:  Function ESBArcTan( X, Y : Extended) : Extended
5385:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5386:  Function ESBArcCos( const X : Extended) : Extended
5387:  Function ESBArcSin( const X : Extended) : Extended
5388:  Function ESBArcSec( const X : Extended) : Extended
5389:  Function ESBArcCosec( const X : Extended) : Extended
5390:  Function ESBLog10( const X : Extended) : Extended
5391:  Function ESBLog2( const X : Extended) : Extended
5392:  Function ESBLogBase( const X, Base : Extended) : Extended
5393:  Function Pow2( const X : Extended) : Extended
5394:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5395:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5396:  Function XtoY( const X, Y : Extended) : Extended
5397:  Function esbTenToY( const Y : Extended) : Extended
5398:  Function esbTwoToY( const Y : Extended) : Extended
5399:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5400:  Function esbISqrt( const I : LongWord) : Longword
5401:  Function ILog2( const I : LongWord) : LongWord
5402:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5403:  Function ESBArCosh( X : Extended) : Extended
5404:  Function ESBArSinh( X : Extended) : Extended
5405:  Function ESBArTanh( X : Extended) : Extended
```

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

```
5490:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5491:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5492:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5493:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5494:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5495:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5496:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5497:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5498:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5499:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5500:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5501:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5502:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5503:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5504:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5505:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5506:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5507:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5508:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5509:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5510:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5511:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5512:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5513:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5514:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5515:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5516:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5517:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5518:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5519:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5520:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5521:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5522:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5523:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5524:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5525: end;
5526:
5527: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5528: begin
5529:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5530:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5531:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5532:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5533:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLns:TJvXPBoundLins;const
      AColor:TColor;Rect:TRect;
5534:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5535:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5536:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool;
5537:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5538:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5539:  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)
5540: end;
5541:
5542:
5543: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5544: begin
5545:  Function StrDec( S : String) : String
5546:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5547:  Function ReadNumFromEdit( Edit : TEdit) : Float
5548:  Procedure WriteNumToFile( var F : Text; X : Float)
5549: end;
5550:
5551: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5552: begin
5553:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5554:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5555:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5556:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5557:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5558:  Procedure TeX_SetGraphTitle( Title : String)
5559:  Procedure TeX_SetOxTitle( Title : String)
5560:  Procedure TeX_SetOyTitle( Title : String)
5561:  Procedure TeX_PlotOxAxis
5562:  Procedure TeX_PlotOyAxis
5563:  Procedure TeX_PlotGrid( Grid : TGrid)
5564:  Procedure TeX_WriteGraphTitle
5565:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5566:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5567:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5568:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5569:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5570:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5571:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
```

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

```
5656:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5657:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5658:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5659:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5660:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5661:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5662:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5663:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5664:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5665:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5666:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5667:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5668:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5669:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
      SubString:ShortString):Bool;
5670:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
      SubStr:ShortString):Bool;
5671:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5672:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
      SubString:ShortString):Bool;
5673:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5674:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5675:  Function ExtractTokensS(const S,
      Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5676:  Function IsChAlphaS( C : Char) : Boolean
5677:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5678:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5679:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5680:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5681:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5682:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5683:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5684:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5685:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5686:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5687:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
      Cardinal):ShortString;
5688:  Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
      Replacements:Cardinal):ShortString;
5689:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5690:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
      Replacements:Cardinal):ShortString
5691:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
      Replacements:Cardinal):ShortString
5692:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5693:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5694:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5695:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5696: end;
5697:
5698:
5699: ********unit uPSI_StUtils; from Systools4*****************************************************
5700: Function SignL( L : LongInt) : Integer
5701: Function SignF( F : Extended) : Integer
5702: Function MinWord( A, B : Word) : Word
5703: Function MidWord( W1, W2, W3 : Word) : Word
5704: Function MaxWord( A, B : Word) : Word
5705: Function MinLong( A, B : LongInt) : LongInt
5706: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5707: Function MaxLong( A, B : LongInt) : LongInt
5708: Function MinFloat( F1, F2 : Extended) : Extended
5709: Function MidFloat( F1, F2, F3 : Extended) : Extended
5710: Function MaxFloat( F1, F2 : Extended) : Extended
5711: Function MakeInteger16( H, L : Byte) : SmallInt
5712: Function MakeWordS( H, L : Byte) : Word
5713: Function SwapNibble( B : Byte) : Byte
5714: Function SwapWord( L : LongInt) : LongInt
5715: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5716: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5717: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5718: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5719: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5720: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5721: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5722: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5723: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5724: Procedure ExchangeBytes( var I, J : Byte)
5725: Procedure ExchangeWords( var I, J : Word)
5726: Procedure ExchangeLongInts( var I, J : LongInt)
5727: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5728: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5729: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5730: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5731: //*****************uPSI_StFIN;*************************************************************
5732: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5733: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
      Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5734: Function BondDuration( Settlement,Maturity:TStDate;Rate,
      Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
```

```
5735: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
      Extended
5736: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5737: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5738: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5739: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5740: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5741: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5742: Function DollarToDecimalText( DecDollar : Extended) : string
5743: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5744: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5745: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5746: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended;
5747: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5748: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5749: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5750: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5751: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5752: Function IsCardValid( const S : string) : Boolean
5753: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5754: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5755: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5756: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5757: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5758: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5759: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5760: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5761: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5762: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5763: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5764: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5765: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5766: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5767: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5768: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5769: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5770: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5771: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5772: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5773:
5774: //*****************************************unit uPSI_StAstroP;
5775: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5776: //*****unit unit uPSI_StStat; Statistic Package of SysTools*****************************************
5777: Function AveDev( const Data : array of Double) : Double
5778: Function AveDev16( const Data, NData : Integer) : Double
5779: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5780: Function Correlation( const Data1, Data2 : array of Double) : Double
5781: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5782: Function Covariance( const Data1, Data2 : array of Double) : Double
5783: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5784: Function DevSq( const Data : array of Double) : Double
5785: Function DevSq16( const Data, NData : Integer) : Double
5786: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5787:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5788: Function GeometricMeanS( const Data : array of Double) : Double
5789: Function GeometricMean16( const Data, NData : Integer) : Double
5790: Function HarmonicMeanS( const Data : array of Double) : Double
5791: Function HarmonicMean16( const Data, NData : Integer) : Double
5792: Function Largest( const Data : array of Double; K : Integer) : Double
5793: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5794: Function MedianS( const Data : array of Double) : Double
5795: Function Median16( const Data, NData : Integer) : Double
5796: Function Mode( const Data : array of Double) : Double
5797: Function Mode16( const Data, NData : Integer) : Double
5798: Function Percentile( const Data : array of Double; K : Double) : Double
5799: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5800: Function PercentRank( const Data : array of Double; X : Double) : Double
5801: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5802: Function Permutations( Number, NumberChosen : Integer) : Extended
5803: Function Combinations( Number, NumberChosen : Integer) : Extended
5804: Function FactorialS( N : Integer) : Extended
5805: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5806: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5807: Function Smallest( const Data : array of Double; K : Integer) : Double
5808: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5809: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5810: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5811:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5812:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5813: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
```

```
5814: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5815: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5816: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5817: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5818: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5819: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5820: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5821: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5822: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5823: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5824: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5825: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5826: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5827: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5828: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5829: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5830: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5831: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5832: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5833: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5834: Function NormSDist( Z : Single) : Single
5835: Function NormSInv( Probability : Single) : Single
5836: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5837: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5838: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5839: Function Erfc( X : Single) : Single
5840: Function GammaLn( X : Single) : Single
5841: Function LargestSort( const Data : array of Double; K : Integer) : Double
5842: Function SmallestSort( const Data : array of double; K : Integer) : Double
5843:
5844: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5845:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5846:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5847:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5848:  Function DefaultMergeName( MergeNum : Integer) : string
5849:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5850:
5851: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5852: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5853:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5854:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5855:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5856:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5857:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5858:  Function LunarPhase( UT : TStDateTimeRec) : Double
5859:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5860:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5861:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5862:  Function FullMoon( D : TStDate) : TStLunarRecord
5863:  Function LastQuarter( D : TStDate) : TStLunarRecord
5864:  Function NewMoon( D : TStDate) : TStLunarRecord
5865:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5866:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5867:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5868:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5869:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5870:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5871:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5872:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5873:  Function SiderealTime( UT : TStDateTimeRec) : Double
5874:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5875:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5876:  Function SEaster( Y, Epoch : Integer) : TStDate
5877:  Function DateTimeToAJD( D : TDateTime) : Double
5878:  Function HoursMin( RA : Double) : ShortString
5879:  Function DegsMin( DC : Double) : ShortString
5880:  Function AJDToDateTime( D : Double) : TDateTime
5881:
5882: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5883:  Function CurrentDate : TStDate
5884:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5885:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5886:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5887:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5888:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5889:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5890:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5891:  Function WeekOfYear( Julian : TStDate) : Byte
5892:  Function AstJulianDate( Julian : TStDate) : Double
5893:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5894:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5895:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5896:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5897:  Function StIsLeapYear( Year : Integer) : Boolean
5898:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5899:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5900:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5901:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
```

```
5902:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5903:  Function CurrentTime : TStTime
5904:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5905:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5906:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5907:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5908:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5909:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5910:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5911:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5912:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5913:  Function StDateToDateTime( D : TStDate) : TDateTime
5914:  Function StTimeToDateTime( T : TStTime) : TDateTime
5915:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5916:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5917:
5918: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5919: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5920:  Function MonthToString( const Month : Integer) : string
5921:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5922:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5923:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5924:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5925:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5926:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5927:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5928:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5929:  Function TimeStringToStTime( const Picture, S : string) : TStTime
5930:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5931:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5932:  Function DateStringIsBlank( const Picture, S : string) : Boolean
5933:  Function InternationalDate( ForceCentury : Boolean) : string
5934:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5935:  Function InternationalTime( ShowSeconds : Boolean) : string
5936:  Procedure ResetInternationalInfo
5937:
5938: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5939:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5940:  Function AnsiUpperCaseShort32( const S : string) : string
5941:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
5942:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
5943:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
5944:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5945:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5946:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5947:  Function Upcase( C : AnsiChar) : AnsiChar
5948:  Function LoCase( C : AnsiChar) : AnsiChar
5949:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
5950:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
5951:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
5952:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5953:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5954:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
5955:  Procedure RaiseContainerError( Code : longint)
5956:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
5957:  Function ProductOverflow( A, B : LongInt) : Boolean
5958:  Function StNewStr( S : string) : PShortString
5959:  Procedure StDisposeStr( PS : PShortString)
5960:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
5961:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
5962:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
5963:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
5964:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
5965:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
5966:
5967: procedure SIRegister_usvd(CL: TPSPascalCompiler);
5968: begin
5969:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
5970:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
5971:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
5972:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
5973:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
5974: end;
5975:
5976: //**********unit unit ; StMath Package of SysTools*****************************************
5977: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
5978: Function PowerS( Base, Exponent : Extended) : Extended
5979: Function StInvCos( X : Double) : Double
5980: Function StInvSin( Y : Double) : Double
5981: Function StInvTan2( X, Y : Double) : Double
5982: Function StTan( A : Double) : Double
5983: Procedure DumpException;  //unit StExpEng;
5984: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
5985:
5986: //**********unit unit ; StCRC Package of SysTools*****************************************
5987: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
5988: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
5989: Function Adler32OfFile( FileName : AnsiString) : LongInt
5990: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
```

```
5991: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
5992: Function Crc16OfFile( FileName : AnsiString) : Cardinal
5993: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
5994: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
5995: Function Crc32OfFile( FileName : AnsiString) : LongInt
5996: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
5997: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
5998: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
5999: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6000: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6001: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6002:
6003: //**********unit unit ; StBCD Package of SysTools*****************************************
6004: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
6005: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
6006: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
6007: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
6008: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
6009: Function NegBcd( const B : TbcdS) : TbcdS
6010: Function AbsBcd( const B : TbcdS) : TbcdS
6011: Function FracBcd( const B : TbcdS) : TbcdS
6012: Function IntBcd( const B : TbcdS) : TbcdS
6013: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6014: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6015: Function ValBcd( const S : string) : TbcdS
6016: Function LongBcd( L : LongInt) : TbcdS
6017: Function ExtBcd( E : Extended) : TbcdS
6018: Function ExpBcd( const B : TbcdS) : TbcdS
6019: Function LnBcd( const B : TbcdS) : TbcdS
6020: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6021: Function PowBcd( const B, E : TbcdS) : TbcdS
6022: Function SqrtBcd( const B : TbcdS) : TbcdS
6023: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6024: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6025: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6026: Function IsIntBcd( const B : TbcdS) : Boolean
6027: Function TruncBcd( const B : TbcdS) : LongInt
6028: Function BcdExt( const B : TbcdS) : Extended
6029: Function RoundBcd( const B : TbcdS) : LongInt
6030: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6031: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6032: Function FormatBcd( const Format : string; const B : TbcdS) : string
6033: Function StrGeneralBcd( const B : TbcdS) : string
6034: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6035: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6036:
6037: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6038: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6039: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6040: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6041: Function StDeEscape( const EscStr : AnsiString) : Char
6042: Function StDoEscape( Delim : Char) : AnsiString
6043: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6044: Function AnsiHashText( const S : string; Size : Integer) : Integer
6045: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6046: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6047: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6048:
6049: //**********unit unit ; StNetCon Package of SysTools*****************************************
6050:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6051:     Constructor Create( AOwner : TComponent)
6052:     Function Connect : DWord
6053:     Function Disconnect : DWord
6054:     RegisterProperty('Password', 'String', iptrw);
6055:     Property('UserName', 'String', iptrw);
6056:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6057:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6058:     Property('LocalDevice', 'String', iptrw);
6059:     Property('ServerName', 'String', iptrw);
6060:     Property('ShareName', 'String', iptrw);
6061:     Property('OnConnect', 'TNotifyEvent', iptrw);
6062:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6063:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6064:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6065:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6066:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6067:   end;
6068: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6069: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6070: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6071: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6072: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6073: Function InitializeCriticalSectionAndSpinCount(var
     lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6074: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6075: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6076: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6077: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6078: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
```

```
6079: Function SuspendThread( hThread : THandle) : DWORD
6080: Function ResumeThread( hThread : THandle) : DWORD
6081: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6082: Function GetCurrentThread : THandle
6083: Procedure ExitThread( dwExitCode : DWORD)
6084: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6085: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6086: Procedure EndThread(ExitCode: Integer);
6087: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6088: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6089: Procedure FreeProcInstance( Proc : FARPROC)
6090: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6091: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6092: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6093: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6094: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6095: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6096: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6097: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6098: Function CurrentParallelJobInfo : TParallelJobInfo
6099: Function ObtainParallelJobInfo : TParallelJobInfo
6100:
6101: ************************************************unit uPSI_JclMime;
6102:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6103:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6104:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6105:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6106:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6107:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6108:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6109:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6110:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6111:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card):
      Cardinal;
6112:
6113: ************************************************unit uPSI_JclPrint;
6114:  Procedure DirectPrint( const Printer, Data : string)
6115:  Procedure SetPrinterPixelsPerInch
6116:  Function GetPrinterResolution : TPoint
6117:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6118:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6119:
6120:
6121: //************************************unit uPSI_ShLwApi;************************************
6122:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6123:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6124:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6125:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6126:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6127:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6128:  Function StrDup( lpSrch : PChar) : PChar
6129:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6130:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6131:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6132:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6133: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6134:  Function StrPBrk( psz, pszSet : PChar) : PChar
6135:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6136:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6137:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6138:  Function StrSpn( psz, pszSet : PChar) : Integer
6139:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6140:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6141:  Function StrToInt( lpSrch : PChar) : Integer
6142:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6143:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6144:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6145:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6146:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6147:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6148:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6149:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6150:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6151:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6152:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6153:  SZ_CONTENTTYPE_HTMLA','String 'text/html
6154:  SZ_CONTENTTYPE_HTMLW','String 'text/html
6155:  SZ_CONTENTTYPE_HTML','string SZ_CONTENTTYPE_HTMLA);
6156:  SZ_CONTENTTYPE_CDFA','String 'application/x-cdf
6157:  SZ_CONTENTTYPE_CDFW','String 'application/x-cdf
6158:  SZ_CONTENTTYPE_CDF','string SZ_CONTENTTYPE_CDFA);
6159:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6160:  STIF_DEFAULT','LongWord( $00000000);
6161:  STIF_SUPPORT_HEX','LongWord( $00000001);
6162:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6163:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6164:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6165:  Function PathAddBackslash( pszPath : PChar) : PChar
```

```
6166:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6167:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6168:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6169:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6170:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6171:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6172:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6173:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6174:  Function PathFileExists( pszPath : PChar) : BOOL
6175:  Function PathFindExtension( pszPath : PChar) : PChar
6176:  Function PathFindFileName( pszPath : PChar) : PChar
6177:  Function PathFindNextComponent( pszPath : PChar) : PChar
6178:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6179:  Function PathGetArgs( pszPath : PChar) : PChar
6180:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6181:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6182:  Function PathGetCharType( ch : Char) : UINT
6183:  GCT_INVALID','LongWord( $0000);
6184:  GCT_LFNCHAR','LongWord( $0001);
6185:  GCT_SHORTCHAR','LongWord( $0002);
6186:  GCT_WILD','LongWord( $0004);
6187:  GCT_SEPARATOR','LongWord( $0008);
6188:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6189:  Function PathIsDirectory( pszPath : PChar) : BOOL
6190:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6191:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6192:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6193:  Function PathIsRelative( pszPath : PChar) : BOOL
6194:  Function PathIsRoot( pszPath : PChar) : BOOL
6195:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6196:  Function PathIsUNC( pszPath : PChar) : BOOL
6197:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6198:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6199:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6200:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6201:  Function PathIsURL( pszPath : PChar) : BOOL
6202:  Function PathMakePretty( pszPath : PChar) : BOOL
6203:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6204:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6205:  Procedure PathQuoteSpaces( lpsz : PChar)
6206:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6207:  Procedure PathRemoveArgs( pszPath : PChar)
6208:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6209:  Procedure PathRemoveBlanks( pszPath : PChar)
6210:  Procedure PathRemoveExtension( pszPath : PChar)
6211:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6212:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6213:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6214:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6215:  Function PathSkipRoot( pszPath : PChar) : PChar
6216:  Procedure PathStripPath( pszPath : PChar)
6217:  Function PathStripToRoot( pszPath : PChar) : BOOL
6218:  Procedure PathUnquoteSpaces( lpsz : PChar)
6219:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6220:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6221:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6222:  Procedure PathUndecorate( pszPath : PChar)
6223:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6224:  URL_SCHEME_INVALID','LongInt'( - 1);
6225:  URL_SCHEME_UNKNOWN','LongInt'( 0);
6226:  URL_SCHEME_FTP','LongInt'( 1);
6227:  URL_SCHEME_HTTP','LongInt'( 2);
6228:  URL_SCHEME_GOPHER','LongInt'( 3);
6229:  URL_SCHEME_MAILTO','LongInt'( 4);
6230:  URL_SCHEME_NEWS','LongInt'( 5);
6231:  URL_SCHEME_NNTP','LongInt'( 6);
6232:  URL_SCHEME_TELNET','LongInt'( 7);
6233:  URL_SCHEME_WAIS','LongInt'( 8);
6234:  URL_SCHEME_FILE','LongInt'( 9);
6235:  URL_SCHEME_MK','LongInt'( 10);
6236:  URL_SCHEME_HTTPS','LongInt'( 11);
6237:  URL_SCHEME_SHELL','LongInt'( 12);
6238:  URL_SCHEME_SNEWS','LongInt'( 13);
6239:  URL_SCHEME_LOCAL','LongInt'( 14);
6240:  URL_SCHEME_JAVASCRIPT','LongInt'( 15);
6241:  URL_SCHEME_VBSCRIPT','LongInt'( 16);
6242:  URL_SCHEME_ABOUT','LongInt'( 17);
6243:  URL_SCHEME_RES','LongInt'( 18);
6244:  URL_SCHEME_MAXVALUE','LongInt'( 19);
6245:  URL_SCHEME', 'Integer
6246:  URL_PART_NONE','LongInt'( 0);
6247:  URL_PART_SCHEME','LongInt'( 1);
6248:  URL_PART_HOSTNAME','LongInt'( 2);
6249:  URL_PART_USERNAME','LongInt'( 3);
6250:  URL_PART_PASSWORD','LongInt'( 4);
6251:  URL_PART_PORT','LongInt'( 5);
6252:  URL_PART_QUERY','LongInt'( 6);
6253:  URL_PART', 'DWORD
6254:  URLIS_URL','LongInt'( 0);
```

```
6255:  URLIS_OPAQUE','LongInt'( 1);
6256:  URLIS_NOHISTORY','LongInt'( 2);
6257:  URLIS_FILEURL','LongInt'( 3);
6258:  URLIS_APPLIABLE','LongInt'( 4);
6259:  URLIS_DIRECTORY','LongInt'( 5);
6260:  URLIS_HASQUERY','LongInt'( 6);
6261:  TUrlIs', 'DWORD
6262:  URL_UNESCAPE','LongWord( $10000000);
6263:  URL_ESCAPE_UNSAFE','LongWord( $20000000);
6264:  URL_PLUGGABLE_PROTOCOL','LongWord( $40000000);
6265:  URL_WININET_COMPATIBILITY','LongWord( DWORD ( $80000000 ));
6266:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord( $02000000);
6267:  URL_ESCAPE_SPACES_ONLY','LongWord( $04000000);
6268:  URL_DONT_SIMPLIFY','LongWord( $08000000);
6269:  URL_NO_META','longword( URL_DONT_SIMPLIFY);
6270:  URL_UNESCAPE_INPLACE','LongWord( $00100000);
6271:  URL_CONVERT_IF_DOSPATH','LongWord( $00200000);
6272:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord( $00400000);
6273:  URL_INTERNAL_PATH','LongWord( $00800000);
6274:  URL_FILE_USE_PATHURL','LongWord( $00010000);
6275:  URL_ESCAPE_PERCENT','LongWord( $00001000);
6276:  URL_ESCAPE_SEGMENT_ONLY','LongWord( $00002000);
6277:  URL_PARTFLAG_KEEPSCHEME','LongWord( $00000001);
6278:  URL_APPLY_DEFAULT','LongWord( $00000001);
6279:  URL_APPLY_GUESSSCHEME','LongWord( $00000002);
6280:  URL_APPLY_GUESSFILE','LongWord( $00000004);
6281:  URL_APPLY_FORCEAPPLY','LongWord( $00000008);
6282:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6283:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6284:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6285:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6286:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6287:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6288:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6289:  Function UrlGetLocation( psz1 : PChar) : PChar
6290:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6291:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6292:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6293:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6294:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6295:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6296:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6297:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6298:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6299:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6300:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6301:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6302:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6303:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6304:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
       pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6305:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6306:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6307:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6308:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6309:  SHREGDEL_DEFAULT','LongWord( $00000000);
6310:  SHREGDEL_HKCU','LongWord( $00000001);
6311:  SHREGDEL_HKLM','LongWord( $00000010);
6312:  SHREGDEL_BOTH','LongWord( $00000011);
6313:  SHREGENUM_DEFAULT','LongWord( $00000000);
6314:  SHREGENUM_HKCU','LongWord( $00000001);
6315:  SHREGENUM_HKLM','LongWord( $00000010);
6316:  SHREGENUM_BOTH','LongWord( $00000011);
6317:  SHREGSET_HKCU','LongWord( $00000001);
6318:  SHREGSET_FORCE_HKCU','LongWord( $00000002);
6319:  SHREGSET_HKLM','LongWord( $00000004);
6320:  SHREGSET_FORCE_HKLM','LongWord( $00000008);
6321:  TSHRegDelFlags', 'DWORD
6322:  TSHRegEnumFlags', 'DWORD
6323:  HUSKEY', 'THandle
6324:  ASSOCF_INIT_NOREMAPCLSID','LongWord( $00000001);
6325:  ASSOCF_INIT_BYEXENAME','LongWord( $00000002);
6326:  ASSOCF_OPEN_BYEXENAME','LongWord( $00000002);
6327:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord( $00000004);
6328:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord( $00000008);
6329:  ASSOCF_NOUSERSETTINGS','LongWord( $00000010);
6330:  ASSOCF_NOTRUNCATE','LongWord( $00000020);
6331:  ASSOCF_VERIFY','LongWord( $00000040);
6332:  ASSOCF_REMAPRUNDLL','LongWord( $00000080);
6333:  ASSOCF_NOFIXUPS','LongWord( $00000100);
6334:  ASSOCF_IGNOREBASECLASS','LongWord( $00000200);
6335:  ASSOCF', 'DWORD
6336:  ASSOCSTR_COMMAND','LongInt'( 1);
6337:  ASSOCSTR_EXECUTABLE','LongInt'( 2);
6338:  ASSOCSTR_FRIENDLYDOCNAME','LongInt'( 3);
6339:  ASSOCSTR_FRIENDLYAPPNAME','LongInt'( 4);
6340:  ASSOCSTR_NOOPEN','LongInt'( 5);
6341:  ASSOCSTR_SHELLNEWVALUE','LongInt'( 6);
6342:  ASSOCSTR_DDECOMMAND','LongInt'( 7);
```

```
6343:  ASSOCSTR_DDEIFEXEC','LongInt'( 8);
6344:  ASSOCSTR_DDEAPPLICATION','LongInt'( 9);
6345:  ASSOCSTR_DDETOPIC','LongInt'( 10);
6346:  ASSOCSTR_INFOTIP','LongInt'( 11);
6347:  ASSOCSTR_MAX','LongInt'( 12);
6348:  ASSOCSTR', 'DWORD
6349:  ASSOCKEY_SHELLEXECCLASS','LongInt'( 1);
6350:  ASSOCKEY_APP','LongInt'( 2);
6351:  ASSOCKEY_CLASS','LongInt'( 3);
6352:  ASSOCKEY_BASECLASS','LongInt'( 4);
6353:  ASSOCKEY_MAX','LongInt'( 5);
6354:  ASSOCKEY', 'DWORD
6355:  ASSOCDATA_MSIDESCRIPTOR','LongInt'( 1);
6356:  ASSOCDATA_NOACTIVATEHANDLER','LongInt'( 2);
6357:  ASSOCDATA_QUERYCLASSSTORE','LongInt'( 3);
6358:  ASSOCDATA_HASPERUSERASSOC','LongInt'( 4);
6359:  ASSOCDATA_MAX','LongInt'( 5);
6360:  ASSOCDATA', 'DWORD
6361:  ASSOCENUM_NONE','LongInt'( 0);
6362:  ASSOCENUM', 'DWORD
6363:  SID_IQueryAssociations','String '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6364:  SHACF_DEFAULT $00000000);
6365:  SHACF_FILESYSTEM','LongWord( $00000001);
6366:  SHACF_URLHISTORY','LongWord( $00000002);
6367:  SHACF_URLMRU','LongWord( $00000004);
6368:  SHACF_USETAB','LongWord( $00000008);
6369:  SHACF_FILESYS_ONLY','LongWord( $00000010);
6370:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord( $10000000);
6371:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord( $20000000);
6372:  SHACF_AUTOAPPEND_FORCE_ON','LongWord( $40000000);
6373:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord( DWORD ( $80000000 ));
6374:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6375:  Procedure SHSetThreadRef( punk : IUnknown)
6376:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6377:  CTF_INSIST','LongWord( $00000001);
6378:  CTF_THREAD_REF','LongWord( $00000002);
6379:  CTF_PROCESS_REF','LongWord( $00000004);
6380:  CTF_COINIT','LongWord( $00000008);
6381:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6382:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6383:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6384:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6385:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6386:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6387:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6388:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6389:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6390:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6391:  Function SetRectEmpty( var lprc : TRect) : BOOL
6392:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6393:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6394:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6395:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6396:
6397:  Function InitializeFlatSB( hWnd : HWND) : Bool
6398:  Procedure UninitializeFlatSB( hWnd : HWND)
6399:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6400:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6401:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6402:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6403:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6404:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6405:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6406:
6407:
6408: //  ****************************************** 204 unit uPSI_ShellAPI;
6409: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6410: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6411: Procedure DragFinish( Drop : HDROP)
6412: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6413: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6414: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6415: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6416: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6417: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6418: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6419: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6420: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6421: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6422: Procedure SHFreeNameMappings( hNameMappings : THandle)
6423:
6424:  DLLVER_PLATFORM_WINDOWS','LongWord( $00000001);
6425:  DLLVER_PLATFORM_NT','LongWord( $00000002);
6426:  DLLVER_MAJOR_MASK','LongWord( Int64 ( $FFFF000000000000 ));
6427:  DLLVER_MINOR_MASK','LongWord( Int64 ( $0000FFFF00000000 ));
6428:  DLLVER_BUILD_MASK','LongWord( Int64 ( $00000000FFFF0000 ));
6429:  DLLVER_QFE_MASK','LongWord( Int64 ( $000000000000FFFF ));
6430:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6431:  Function SimpleXMLEncode( const S : string) : string
```

```
6432:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6433:  Function XMLEncode( const S : string) : string
6434:  Function XMLDecode( const S : string) : string
6435:  Function EntityEncode( const S : string) : string
6436:  Function EntityDecode( const S : string) : string
6437:
6438: procedure RIRegister_CPort_Routines(S: TPSExec);
6439:  Procedure EnumComPorts( Ports : TStrings)
6440:  Procedure ListComPorts( Ports : TStrings)
6441:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6442:  Function GetComPorts: TStringlist;
6443:  Function StrToBaudRate( Str : string) : TBaudRate
6444:  Function StrToStopBits( Str : string) : TStopBits
6445:  Function StrToDataBits( Str : string) : TDataBits
6446:  Function StrToParity( Str : string) : TParityBits
6447:  Function StrToFlowControl( Str : string) : TFlowControl
6448:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6449:  Function StopBitsToStr( StopBits : TStopBits) : string
6450:  Function DataBitsToStr( DataBits : TDataBits) : string
6451:  Function ParityToStr( Parity : TParityBits) : string
6452:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6453:  Function ComErrorsToStr( Errors : TComErrors) : String
6454:
6455:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6456:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6457:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6458:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6459:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6460:  Function GetMessagePos : DWORD
6461:  Function GetMessageTime : Longint
6462:  Function GetMessageExtraInfo : Longint
6463:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6464:  Procedure JAddToRecentDocs( const Filename : string)
6465:  Procedure ClearRecentDocs
6466:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6467:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6468:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6469:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6470:  Function RecycleFile( FileToRecycle : string) : Boolean
6471:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6472:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6473:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6474:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6475:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6476:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6477:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6478:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6479:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6480:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6481:
6482: ****************************************** unit uPSI_JclPeImage;
6483:
6484:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6485: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6486:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6487:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
6488:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6489:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6490:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6491:  Function PeDoesExportFunction(const FileName:TFileName;const
      FuncName:string;Options:TJclSmartCompOptions):Bool;
6492:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6493:  Function PeIsExportFunctionForwarded(const FileName:TFileName;const
      FunctionName:string;Options:TJclSmartCompOptions):Bool
6494:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean
6495:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
6496:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean
6497:  Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const
      LibraryName:string; IncludeLibNames : Boolean): Boolean
6498:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6499:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6500:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6501:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
      NamesList:TStrings):Bool
6502:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
```

```
6503:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
      Descript:Bool):Bool;
6504:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6505:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6506:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6507:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6508:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6509:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6510:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
      PImageSectionHeader
6511:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6512:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6513:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
      ___Pointer;
6514:   SIRegister_TJclPeSectionStream(CL);
6515:   SIRegister_TJclPeMapImgHookItem(CL);
6516:   SIRegister_TJclPeMapImgHooks(CL);
6517:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
      NtHeaders:TImageNtHeaders):Boolean
6518:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6519:   Type TJclBorUmSymbolKind', '(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6520:   TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6521:   TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6522:   TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6523:   TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6524:   TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6525:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
      TJclBorUmDescription) : Integer) : TJclBorUmResult;
6526:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
      Descript:TJclBorUmDescription):TJclBorUmResult;
6527:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6528:  Function PeBorUnmangleName3( const Name : string) : string;
6529:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6530:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6531:
6532:
6533: //***************** SysTools uPSI_StSystem; ************************************
6534:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6535:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6536:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6537:  //Procedure EnumerateDirectories(const
      StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6538:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
      IncludeItem:TIncludeItemFunc);
6539:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6540:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6541:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6542:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6543:  Function FlushOsBuffers( Handle : Integer) : Boolean
6544:  Function GetCurrentUser : AnsiString
6545:  Function GetDiskClass( Drive : Char) : DiskClass
6546:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
      SectorsPerCluster:Cardinal):Bool;
6547:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
      DiskSize:Double):Bool;
6548:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
      DiskSize:Comp):Boolean;
6549:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6550:  Function getDiskSpace2(const path: String; index: integer): int64;
6551:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6552:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6553:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6554:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6555:  Function GetLongPath( const APath : AnsiString) : AnsiString
6556:  Function GetMachineName : AnsiString
6557:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6558:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6559:  Function GetShortPath( const APath : AnsiString) : AnsiString
6560:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6561:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6562:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6563:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6564:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6565:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6566:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6567:  Function IsDriveReady( Drive : Char) : Boolean
6568:  Function IsFile( const FileName : AnsiString) : Boolean
6569:  Function IsFileArchive( const S : AnsiString) : Integer
6570:  Function IsFileHidden( const S : AnsiString) : Integer
6571:  Function IsFileReadOnly( const S : AnsiString) : Integer
6572:  Function IsFileSystem( const S : AnsiString) : Integer
6573:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6574:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6575:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6576:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6577:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6578:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6579:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6580:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
```

```
6581:  Function ValidDrive( Drive : Char) : Boolean
6582:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6583:
6584: //****************************unit uPSI_JclLANMan;*****************************************
6585: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6586:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean
6587:  Function DeleteAccount( const Servername, Username : string) : Boolean
6588:  Function DeleteLocalAccount( Username : string) : Boolean
6589:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6590:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6591:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6592:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6593:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6594:  Function LocalGroupExists( const Group : string) : Boolean
6595:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6596:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6597:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6598:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6599:  Function IsLocalAccount( const AccountName : string) : Boolean
6600:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6601:  Function GetRandomString( NumChar : cardinal) : string
6602:
6603: //****************************unit uPSI_cUtils;*****************************************
6604: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6605: Function cIsWinNT : boolean
6606:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6607:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6608:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6609:  Function cGetShortName( FileName : string) : string
6610:  Procedure cShowError( Msg : String)
6611:  Function cCommaStrToStr( s : string; formatstr : string) : string
6612:  Function cIncludeQuoteIfSpaces( s : string) : string
6613:  Function cIncludeQuoteIfNeeded( s : string) : string
6614:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6615:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6616:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6617:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6618:  Function cCodeInstoStr( s : string) : string
6619:  Function cStrtoCodeIns( s : string) : string
6620:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6621:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6622:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6623:  Function cPointtoStr( const pt : TPoint) : string
6624:  Function cListtoStr( const List : TStrings) : string
6625:  Function ListtoStr( const List : TStrings) : string
6626:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6627:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6628:  Function cGetFileTyp( const FileName : string) : TUnitType
6629:  Function cGetExTyp( const FileName : string) : TExUnitType
6630:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6631:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6632:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6633:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6634:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6635:  Function cGenMakePath( FileName : String) : String;
6636:  Function cGenMakePath2( FileName : String) : String
6637:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6638:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6639:  Function cCalcMod( Count : Integer) : Integer
6640:  Function cGetVersionString( FileName : string) : string
6641:  Function cCheckChangeDir( var Dir : string) : boolean
6642:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6643:  Function cIsNumeric( s : string) : boolean
6644:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6645:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6646:  Function GetFileTyp( const FileName : string) : TUnitType
6647:  Function Atoi(const aStr: string): integer
6648:  Function Itoa(const aint: integer): string
6649:
6650:
6651: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6652: begin
6653:   FindClass('TOBJECT'),'EHTTP
6654:   FindClass('TOBJECT'),'EHTTPParser
6655:   //AnsiCharSet', 'set of AnsiChar
6656:   AnsiStringArray', 'array of AnsiString
6657:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6658:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6659:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6660:    +'TPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6661:    +'CustomMinVersion : Integer; end
6662:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6663:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6664:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6665:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
```

```
6666:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6667:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6668:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6669:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6670:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6671:    +'nection, hntOrigin, hntKeepAlive )
6672:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6673:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6674:    +' AnsiString; end
6675:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6676:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6677:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6678:   //PHTTPContentLength', '^THTTPContentLength // will not work
6679:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6680:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6681:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6682:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6683:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6684:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6685:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6686:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6687:    +' CustomStr : AnsiString; end
6688:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6689:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6690:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6691:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6692:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6693:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6694:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6695:    +'m; Custom : AnsiString; end
6696:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6697:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6698:    +' Custom : AnsiString; end
6699:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6700:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6701:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6702:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6703:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6704:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6705:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6706:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6707:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6708:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6709:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6710:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6711:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6712:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6713:    +'FieldEnum; List : array of THTTPContentEncoding; end
6714:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6715:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6716:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6717:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6718:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6719:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6720:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6721:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6722:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6723:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6724:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6725:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6726:    +'CustomFieldArray; Custom : AnsiString; end
6727:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6728:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6729:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6730:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6731:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6732:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6733:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6734:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6735:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6736:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6737:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6738:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6739:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6740:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6741:   THTTPFixedHeaders','array[0..42] of AnsiString
6742:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6743:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6744:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6745:   THTTPRequestStartLine','record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6746:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6747:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6748:    +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;IfUnmodifiedSince:THTTPDateField;end
6749:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6750:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6751:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6752:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6753:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6754:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
```

```
6755:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6756:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6757:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6758:    +' THTTPDateField; Age : THTTPAgeField; end
6759:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6760:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6761:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6762:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6763:  Procedure InitHTTPRequest( var A : THTTPRequest)
6764:  Procedure InitHTTPResponse( var A : THTTPResponse)
6765:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6766:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6767:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6768:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6769:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6770:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6771:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6772:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6773:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6774:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6775:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6776:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6777:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6778:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6779:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6780:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6781:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6782:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6783:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6784:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6785:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6786:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6787:   THTTPStringOption', '( hsoNone )
6788:   THTTPStringOptions', 'set of THTTPStringOption
6789:   FindClass('TOBJECT'),'TAnsiStringBuilder
6790:
6791:  Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6792:  Procedure BuildStrHTTPContentLengthValue(const
      A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6793:  Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
      B:TAnsiStringBuilder;P:THTTPStringOptions)
6794:  Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
      P:THTTPStringOptions)
6795:  Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6796:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6797:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6798:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6799:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6800:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6801:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6802:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6803:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6804:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6805:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6806:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
6807:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6808:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6809:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions)
6810:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6811:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6812:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6813:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6814:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6815:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6816:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6817:  Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
```

```
6818:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6819:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
      THTTPStrOptions);
6820:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6821:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
      P:THTTPStringOptions);
6822:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6823:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6824:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6825:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6826:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6827:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6828:  Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
      Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6829:    +PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6830:   SIRegister_THTTPParser(CL);
6831:   FindClass('TOBJECT'),'THTTPContentDecoder
6832:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6833:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6834:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6835:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6836:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6837:   SIRegister_THTTPContentDecoder(CL);
6838:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6839:   FindClass('TOBJECT'),'THTTPContentReader
6840:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6841:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
      LogLevel:Int;
6842:   SIRegister_THTTPContentReader(CL);
6843:   THTTPContentWriterMechanism','(hctmEvent, hctmString, hctmStream, hctmFile )
6844:   FindClass('TOBJECT'),'THTTPContentWriter
6845:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6846:   SIRegister_THTTPContentWriter(CL);
6847:  Procedure SelfTestcHTTPUtils
6848: end;
6849:
6850: (*----------------------------------------------------------------------------*)
6851: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6852: begin
6853:  'TLSLibraryVersion','String '1.00
6854:  'TLSError_None','LongInt'( 0);
6855:  'TLSError_InvalidBuffer','LongInt'( 1);
6856:  'TLSError_InvalidParameter','LongInt'( 2);
6857:  'TLSError_InvalidCertificate','LongInt'( 3);
6858:  'TLSError_InvalidState','LongInt'( 4);
6859:  'TLSError_DecodeError','LongInt'( 5);
6860:  'TLSError_BadProtocol','LongInt'( 6);
6861:  Function TLSErrorMessage( const TLSError : Integer) : String
6862:   SIRegister_ETLSError(CL);
6863:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6864:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6865:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6866:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6867:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6868:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6869:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6870:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6871:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6872:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6873:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6874:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6875:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6876:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6877:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6878:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6879:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6880:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6881:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6882:   PTLSRandom', '^TTLSRandom // will not work
6883:  Procedure InitTLSRandom( var Random : TTLSRandom)
6884:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6885:  'TLSSessionIDMaxLen','LongInt'( 32);
6886:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6887:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6888:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6889:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6890:    +'; Signature : TTLSSignatureAlgorithm; end
6891:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6892:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6893:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6894:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6895:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6896:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6897:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6898:    +'nteger; Supported : Boolean; end
6899:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6900:  'TLS_MAC_MAXDIGESTSIZE','LongInt'( 64);
```

```
6901:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6902:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6903:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6904:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6905:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6906:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6907:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6908:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6909:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
      Size:Int):AString;
6910:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
      Size:Integer):AnsiString
6911:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6912:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
      Size:Int):AnsiString;
6913:  Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
      ClientRandom : AnsiString; const Size : Integer) : AnsiString
6914:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6915:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6916:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6917:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
      ServerRandom:AnsiString) : AnsiString
6918:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6919:    +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6920:    +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6921:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
      IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6922:  Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
      ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6923:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt'( 16384 - 1);
6924:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024);
6925:  Procedure SelfTestcTLSUtils
6926: end;
6927:
6928: (*----------------------------------------------------------------------------*)
6929: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6930: begin
6931:   sPosData','record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
      integer; disks : integer; mx : integer; my : integer; end
6932:  // pBoard', '^tBoard // will not work
6933:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6934:  Function rCheckMove( color : byte; cx, cy : integer) : integer
6935:  //Function rDoStep( data : pBoard) : word
6936:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6937: end;
6938:
6939: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6940: begin
6941: Function InEditMode( ADataset : TDataset) : Boolean
6942: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6943: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6944: Function GetFieldText( AField : TField) : String
6945: end;
6946:
6947: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6948: begin
6949:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6950:   TMyPrintRange', '( prAll, prSelected )
6951:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6952:    +'ded, ssDateTime, ssTime, ssCustom )
6953:   TSortDirection', '( sdAscending, sdDescending )
6954:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6955:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
6956:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6957:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6958:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
6959:   SIRegister_TSortOptions(CL);
6960:   SIRegister_TPrintOptions(CL);
6961:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6962:   SIRegister_TSortedList(CL);
6963:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6964:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6965:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6966:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
6967:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
6968:   SIRegister_TFontSetting(CL);
6969:   SIRegister_TFontList(CL);
6970:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
6971:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
6972:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6973:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6974:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
6975:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
6976:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
6977:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
6978:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
6979:    +'r; var SortStyle : TSortStyle)
6980:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
```

```
6981:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
6982:   SIRegister_TSortGrid(CL);
6983:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
6984:  Function NormalCompare( const Str1, Str2 : String) : Integer
6985:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
6986:  Function NumericCompare( const Str1, Str2 : String) : Integer
6987:  Function TimeCompare( const Str1, Str2 : String) : Integer
6988:  //Function Compare( Item1, Item2 : Pointer) : Integer
6989: end;
6990:
6991: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
6992:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
6993:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
6994:  Procedure IBDataBaseError
6995:  Function StatusVector : PISC_STATUS
6996:  Function StatusVectorArray : PStatusVector
6997:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
6998:  Function StatusVectorAsText : string
6999:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7000:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7001:
7002:
7003: //****************************unit uPSI_BoldUtils;****************************************
7004:  Function CharCount( c : char; const s : string) : integer
7005:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
7006:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7007:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7008:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7009:  Function BoldTrim( const S : string) : string
7010:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
7011:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7012:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7013:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7014:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7015:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7016:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7017:  Function CapitalisedToSpaced( Capitalised : String) : String
7018:  Function SpacedToCapitalised( Spaced : String) : String
7019:  Function BooleanToString( BoolValue : Boolean) : String
7020:  Function StringToBoolean( StrValue : String) : Boolean
7021:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7022:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7023:  Function StringListToVarArray( List : TStringList) : variant
7024:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7025:  Function GetComputerNameStr : string
7026:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7027:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7028:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7029:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7030:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7031:  Procedure EnsureTrailing( var Str : String; ch : char)
7032:  Function BoldDirectoryExists( const Name : string) : Boolean
7033:  Function BoldForceDirectories( Dir : string) : Boolean
7034:  Function BoldRootRegistryKey : string
7035:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7036:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7037:  Function LogicalAnd( A, B : Integer) : Boolean
7038:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7039:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7040:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7041:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7042:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7043:  Function IsFirstInstance : Boolean
7044:  Procedure ActivateFirst( AString : PChar)
7045:  Procedure ActivateFirstCommandLine
7046:  function MakeAckPkt(const BlockNumber: Word): string;
7047:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string;
7048:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7049:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7050:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7051:  function IdStrToWord(const Value: String): Word;
7052:  function IdWordToStr(const Value: Word): WordStr;
7053:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7054:  Function CPUFeatures : TCPUFeatures
7055:
7056: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7057: begin
7058:   AddTypeS('TXRTLBitIndex', 'Integer
7059:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7060:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7061:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7062:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7063:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7064:  Function XRTLSwapHiLo16( X : Word) : Word
7065:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7066:  Function XRTLSwapHiLo64( X : Int64) : Int64
7067:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7068:  Function XRTLROR32( A, S : Cardinal) : Cardinal
```

```
7069:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7070:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7071:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7072:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7073:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7074:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7075:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7076:  Function XRTLPopulation( A : Cardinal) : Cardinal
7077: end;
7078:
7079: Function XRTLURLDecode( const ASrc : WideString) : WideString
7080: Function XRTLURLEncode( const ASrc : WideString) : WideString
7081: Function XRTLURINormalize( const AURI : WideString) : WideString
7082: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7083: Function XRTLExtractLongPathName(APath: string): string;
7084:
7085: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7086: begin
7087:   AddTypeS('Int8', 'ShortInt
7088:   AddTypeS('Int16', 'SmallInt
7089:   AddTypeS('Int32', 'LongInt
7090:   AddTypeS('UInt8', 'Byte
7091:   AddTypeS('UInt16', 'Word
7092:   AddTypeS('UInt32', 'LongWord
7093:   AddTypeS('UInt64', 'Int64
7094:   AddTypeS('Word8', 'UInt8
7095:   AddTypeS('Word16', 'UInt16
7096:   AddTypeS('Word32', 'UInt32
7097:   AddTypeS('Word64', 'UInt64
7098:   AddTypeS('LargeInt', 'Int64
7099:   AddTypeS('NativeInt', 'Integer
7100:   AddTypeS('NativeUInt', 'Cardinal
7101:   Const('BitsPerByte','LongInt'( 8);
7102:   Const('BitsPerWord','LongInt'( 16);
7103:   Const('BitsPerLongWord','LongInt'( 32);
7104:  //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8);
7105:  //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8);
7106:  Function MinI( const A, B : Integer) : Integer
7107:  Function MaxI( const A, B : Integer) : Integer
7108:  Function MinC( const A, B : Cardinal) : Cardinal
7109:  Function MaxC( const A, B : Cardinal) : Cardinal
7110:  Function SumClipI( const A, I : Integer) : Integer
7111:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7112:  Function InByteRange( const A : Int64) : Boolean
7113:  Function InWordRange( const A : Int64) : Boolean
7114:  Function InLongWordRange( const A : Int64) : Boolean
7115:  Function InShortIntRange( const A : Int64) : Boolean
7116:  Function InSmallIntRange( const A : Int64) : Boolean
7117:  Function InLongIntRange( const A : Int64) : Boolean
7118:   AddTypeS('Bool8', 'ByteBool
7119:   AddTypeS('Bool16', 'WordBool
7120:   AddTypeS('Bool32', 'LongBool
7121:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7122:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7123:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7124:  Const('MinSingle','Single').setExtended( 1.5E-45);
7125:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7126:  Const('MinDouble','Double').setExtended( 5.0E-324);
7127:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7128:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7129:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7130:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7131:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7132:  Function MinF( const A, B : Float) : Float
7133:  Function MaxF( const A, B : Float) : Float
7134:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7135:  Function InSingleRange( const A : Float) : Boolean
7136:  Function InDoubleRange( const A : Float) : Boolean
7137:  Function InCurrencyRange( const A : Float) : Boolean;
7138:  Function InCurrencyRange1( const A : Int64) : Boolean;
7139:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7140:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7141:  Function FloatIsInfinity( const A : Extended) : Boolean
7142:  Function FloatIsNaN( const A : Extended) : Boolean
7143:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7144:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7145:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7146:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7147:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7148:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7149:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7150:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7151:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7152:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7153:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7154:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7155:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7156:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
```

```
7157:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7158:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7159:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7160:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7161:  Function cIsHighBitSet( const Value : LongWord) : Boolean
7162:  Function SetBitScanForward( const Value : LongWord) : Integer;
7163:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7164:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7165:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7166:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7167:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7168:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7169:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7170:  Function cReverseBits( const Value : LongWord) : LongWord;
7171:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7172:  Function cSwapEndian( const Value : LongWord) : LongWord
7173:  Function cTwosComplement( const Value : LongWord) : LongWord
7174:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7175:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7176:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7177:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7178:  Function cBitCount( const Value : LongWord) : LongWord
7179:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7180:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7181:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7182:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7183:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7184:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7185:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7186:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7187:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7188: //  AddTypeS('CharSet', 'set of AnsiChar
7189:   AddTypeS('CharSet', 'set of Char       //!!!
7190:   AddTypeS('AnsiCharSet', 'TCharSet
7191:   AddTypeS('ByteSet', 'set of Byte
7192:   AddTypeS('AnsiChar', 'Char
7193:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7194:  Function AsByteSet( const C : array of Byte) : ByteSet
7195:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7196:  Procedure ClearCharSet( var C : CharSet)
7197:  Procedure FillCharSet( var C : CharSet)
7198:  Procedure ComplementCharSet( var C : CharSet)
7199:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7200:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7201:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7202:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7203:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7204:  Function IsSubSet( const A, B : CharSet) : Boolean
7205:  Function IsEqual( const A, B : CharSet) : Boolean
7206:  Function IsEmpty( const C : CharSet) : Boolean
7207:  Function IsComplete( const C : CharSet) : Boolean
7208:  Function cCharCount( const C : CharSet) : Integer
7209:  Procedure ConvertCaseInsensitive( var C : CharSet)
7210:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7211:  Function IntRangeLength( const Low, High : Integer) : Int64
7212:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7213:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7214:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7215:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7216:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7217:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7218:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7219:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7220:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7221:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7222:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7223:   AddTypeS('UnicodeChar', 'WideChar
7224:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7225:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7226:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7227:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7228:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7229:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7230:  Function cSgn( const A : LongInt) : Integer;
7231:  Function cSgn1( const A : Int64) : Integer;
7232:  Function cSgn2( const A : Extended) : Integer;
7233:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7234:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7235:  Function WideCharToInt( const A : WideChar) : Integer
7236:  Function CharToInt( const A : Char) : Integer
7237:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7238:  Function IntToWideChar( const A : Integer) : WideChar
7239:  Function IntToChar( const A : Integer) : Char
7240:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7241:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7242:  Function IsHexChar( const Ch : Char) : Boolean
7243:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7244:  Function HexWideCharToInt( const A : WideChar) : Integer
7245:  Function HexCharToInt( const A : Char) : Integer
```

```
7246:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7247:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7248:  Function IntToUpperHexChar( const A : Integer) : Char
7249:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7250:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
7251:  Function IntToLowerHexChar( const A : Integer) : Char
7252:  Function IntToStringA( const A : Int64) : AnsiString
7253:  Function IntToStringW( const A : Int64) : WideString
7254:  Function IntToString( const A : Int64) : String
7255:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7256:  Function UIntToStringW( const A : NativeUInt) : WideString
7257:  Function UIntToString( const A : NativeUInt) : String
7258:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7259:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7260:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7261:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7262:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7263:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7264:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7265:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7266:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7267:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7268:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7269:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7270:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7271:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7272:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7273:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7274:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7275:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7276:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7277:  Function StringToInt64A( const S : AnsiString) : Int64
7278:  Function StringToInt64W( const S : WideString) : Int64
7279:  Function StringToInt64( const S : String) : Int64
7280:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7281:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7282:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7283:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7284:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7285:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7286:  Function StringToIntA( const S : AnsiString) : Integer
7287:  Function StringToIntW( const S : WideString) : Integer
7288:  Function StringToInt( const S : String) : Integer
7289:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7290:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7291:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7292:  Function StringToLongWordA( const S : AnsiString) : LongWord
7293:  Function StringToLongWordW( const S : WideString) : LongWord
7294:  Function StringToLongWord( const S : String) : LongWord
7295:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7296:  Function HexToUIntW( const S : WideString) : NativeUInt
7297:  Function HexToUInt( const S : String) : NativeUInt
7298:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7299:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7300:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7301:  Function HexToLongWordA( const S : AnsiString) : LongWord
7302:  Function HexToLongWordW( const S : WideString) : LongWord
7303:  Function HexToLongWord( const S : String) : LongWord
7304:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7305:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7306:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7307:  Function OctToLongWordA( const S : AnsiString) : LongWord
7308:  Function OctToLongWordW( const S : WideString) : LongWord
7309:  Function OctToLongWord( const S : String) : LongWord
7310:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7311:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7312:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7313:  Function BinToLongWordA( const S : AnsiString) : LongWord
7314:  Function BinToLongWordW( const S : WideString) : LongWord
7315:  Function BinToLongWord( const S : String) : LongWord
7316:  Function FloatToStringA( const A : Extended) : AnsiString
7317:  Function FloatToStringW( const A : Extended) : WideString
7318:  Function FloatToString( const A : Extended) : String
7319:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7320:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7321:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7322:  Function StringToFloatA( const A : AnsiString) : Extended
7323:  Function StringToFloatW( const A : WideString) : Extended
7324:  Function StringToFloat( const A : String) : Extended
7325:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7326:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7327:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7328:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const
      PadChar: AnsiChar) : AnsiString
7329:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7330:  unit uPSI_cFundamentUtils;
7331:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7332:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7333:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
```

```
7334:  Const('CCHARSET','Stringb64_XXEncode);
7335:  Const('CHEXSET','String'0123456789ABCDEF
7336:  Const('HEXDIGITS','String'0123456789ABCDEF
7337:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7338:  Const('DIGISET','String'0123456789
7339:  Const('LETTERSET','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7340:  Const('DIGISET2','TCharset').SetSet('0123456789'
7341:  Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7342:  Const('HEXSET2','TCharset').SetSET('0123456789ABCDEF');
7343:  Const('NUMBERSET','TCharset').SetSet('0123456789');
7344:  Const('NUMBERS','String'0123456789');
7345:  Const('LETTERS','String'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7346:  Function CharSetToStr( const C : CharSet) : AnsiString
7347:  Function StrToCharSet( const S : AnsiString) : CharSet
7348:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7349:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7350:  Function UUDecode( const S : AnsiString) : AnsiString
7351:  Function XXDecode( const S : AnsiString) : AnsiString
7352:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7353:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7354:  Function InterfaceToStrW( const I : IInterface) : WideString
7355:  Function InterfaceToStr( const I : IInterface) : String
7356:  Function ObjectClassName( const O : TObject) : String
7357:  Function ClassClassName( const C : TClass) : String
7358:  Function ObjectToStr( const O : TObject) : String
7359:  Function ObjectToString( const O : TObject) : String
7360:  Function CharSetToStr( const C : CharSet) : AnsiString
7361:  Function StrToCharSet( const S : AnsiString) : CharSet
7362:  Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7363:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7364:  Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
      : Boolean; const Slots : LongWord) : LongWord
7365:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7366:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7367:  Const('Bytes1KB','LongInt'( 1024);
7368:   SIRegister_IInterface(CL);
7369:  Procedure SelfTestCFundamentUtils
7370:
7371: Function CreateSchedule : IJclSchedule
7372: Function NullStamp : TTimeStamp
7373: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7374: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7375: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7376:
7377: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7378: begin
7379:  AddTypeS('TFunc', 'function(X : Float) : Float;
7380: Function InitGraphics( Width, Height : Integer) : Boolean
7381: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7382: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7383: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7384: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7385: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7386: Procedure SetGraphTitle( Title : String)
7387: Procedure SetOxTitle( Title : String)
7388: Procedure SetOyTitle( Title : String)
7389: Function GetGraphTitle : String
7390: Function GetOxTitle : String
7391: Function GetOyTitle : String
7392: Procedure PlotOxAxis( Canvas : TCanvas)
7393: Procedure PlotOyAxis( Canvas : TCanvas)
7394: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7395: Procedure WriteGraphTitle( Canvas : TCanvas)
7396: Function SetMaxCurv( NCurv : Byte) : Boolean
7397: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7398: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7399: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7400: Procedure SetCurvStep( CurvIndex, Step : Integer)
7401: Function GetMaxCurv : Byte
7402: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7403: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7404: Function GetCurvLegend( CurvIndex : Integer) : String
7405: Function GetCurvStep( CurvIndex : Integer) : Integer
7406: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7407: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7408: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7409: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7410: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7411: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7412: Function Xpixel( X : Float) : Integer
7413: Function Ypixel( Y : Float) : Integer
7414: Function Xuser( X : Integer) : Float
7415: Function Yuser( Y : Integer) : Float
7416: end;
7417:
7418: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7419: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
```

```
7420: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7421: Procedure FFT_Integer_Cleanup
7422: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7423: //unit uPSI_JclStreams;
7424: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7425: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7426: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7427: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7428:
7429: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7430: begin
7431:  FindClass('TOBJECT'),'EInvalidDest
7432:  FindClass('TOBJECT'),'EFCantMove
7433:  Procedure fmxCopyFile( const FileName, DestName : string)
7434:  Procedure fmxMoveFile( const FileName, DestName : string)
7435:  Function fmxGetFileSize( const FileName : string) : LongInt
7436:  Function fmxFileDateTime( const FileName : string) : TDateTime
7437:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7438:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7439: end;
7440:
7441: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7442: begin
7443:  SIRegister_IFindFileIterator(CL);
7444:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7445: end;
7446:
7447: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7448: begin
7449:  Function SkipWhite( cp : PChar) : PChar
7450:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7451:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7452:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7453: end;
7454:
7455: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7456: begin
7457:   SIRegister_TStringHashMapTraits(CL);
7458:  Function CaseSensitiveTraits : TStringHashMapTraits
7459:  Function CaseInsensitiveTraits : TStringHashMapTraits
7460:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7461:    +'e; Right : PHashNode; end
7462:   //PHashArray', '^THashArray // will not work
7463:   SIRegister_TStringHashMap(CL);
7464:  THashValue', 'Cardinal
7465:  Function StrHash( const s : string) : THashValue
7466:  Function TextHash( const s : string) : THashValue
7467:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7468:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7469:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7470:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7471:   SIRegister_TCaseSensitiveTraits(CL);
7472:   SIRegister_TCaseInsensitiveTraits(CL);
7473:
7474:
7475: //************************************************************************unit uPSI_umath;
7476:  Function uExpo( X : Float) : Float
7477:  Function uExp2( X : Float) : Float
7478:  Function uExp10( X : Float) : Float
7479:  Function uLog( X : Float) : Float
7480:  Function uLog2( X : Float) : Float
7481:  Function uLog10( X : Float) : Float
7482:  Function uLogA( X, A : Float) : Float
7483:  Function uIntPower( X : Float; N : Integer): Float
7484:  Function uPower( X, Y : Float) : Float
7485:  Function SgnGamma( X : Float) : Integer
7486:  Function Stirling( X : Float) : Float
7487:  Function StirLog( X : Float) : Float
7488:  Function Gamma( X : Float) : Float
7489:  Function LnGamma( X : Float) : Float
7490:  Function DiGamma( X : Float) : Float
7491:  Function TriGamma( X : Float) : Float
7492:  Function IGamma( X : Float) : Float
7493:  Function JGamma( X : Float) : Float
7494:  Function InvGamma( X : Float) : Float
7495:  Function Erf( X : Float) : Float
7496:  Function Erfc( X : Float) : Float
7497: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7498: { Correlation coefficient between samples X and Y }
7499: function DBeta(A, B, X : Float) : Float;
7500: { Density of Beta distribution with parameters A and B }
7501: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7502: Function Beta(X, Y : Float) : Float
7503: Function Binomial( N, K : Integer) : Float
7504: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7505: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7506: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7507: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7508: Function DNorm( X : Float) : Float
```

```
7509:
7510: function DGamma(A, B, X : Float) : Float;
7511: { Density of Gamma distribution with parameters A and B }
7512: function DKhi2(Nu : Integer; X : Float) : Float;
7513: { Density of Khi-2 distribution with Nu d.o.f. }
7514: function DStudent(Nu : Integer; X : Float) : Float;
7515: { Density of Student distribution with Nu d.o.f. }
7516: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7517: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7518: function IBeta(A, B, X : Float) : Float;
7519: { Incomplete Beta function}
7520: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7521:
7522: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7523: begin
7524:  Procedure SetOptAlgo( Algo : TOptAlgo)
7525: procedure SetOptAlgo(Algo : TOptAlgo);
7526: { ----------------------------------------------------------------
7527:   Sets the optimization algorithm according to Algo, which must be
7528:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7529:
7530:  Function GetOptAlgo : TOptAlgo
7531:  Procedure SetMaxParam( N : Byte)
7532:  Function GetMaxParam : Byte
7533:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7534:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7535:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7536:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
      Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7537:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
      MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7538:  Procedure SetMCFile( FileName : String)
7539:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7540:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
      LastPar:Integer;V:TMatrix);
7541: end;
7542:
7543: (*----------------------------------------------------------------------------*)
7544: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7545: begin
7546:  Procedure SaveSimplex( FileName : string)
7547:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7548: end;
7549: (*----------------------------------------------------------------------------*)
7550: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7551: begin
7552:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7553:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7554: end;
7555:
7556: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7557: begin
7558:  Function LTrim( S : String) : String
7559:  Function RTrim( S : String) : String
7560:  Function uTrim( S : String) : String
7561:  Function StrChar( N : Byte; C : Char) : String
7562:  Function RFill( S : String; L : Byte) : String
7563:  Function LFill( S : String; L : Byte) : String
7564:  Function CFill( S : String; L : Byte) : String
7565:  Function Replace( S : String; C1, C2 : Char) : String
7566:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7567:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7568:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7569:  Function FloatStr( X : Float) : String
7570:  Function IntStr( N : LongInt) : String
7571:  Function uCompStr( Z : Complex) : String
7572: end;
7573:
7574: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7575: begin
7576:  Function uSinh( X : Float) : Float
7577:  Function uCosh( X : Float) : Float
7578:  Function uTanh( X : Float) : Float
7579:  Function uArcSinh( X : Float) : Float
7580:  Function uArcCosh( X : Float) : Float
7581:  Function ArcTanh( X : Float) : Float
7582:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7583: end;
7584:
7585: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7586: begin
7587: type RNG_Type =
7588:   (RNG_MWC,      { Multiply-With-Carry }
7589:    RNG_MT,       { Mersenne Twister }
7590:    RNG_UVAG);    { Universal Virtual Array Generator }
7591:  Procedure SetRNG( RNG : RNG_Type)
7592:  Procedure InitGen( Seed : RNG_IntType)
7593:  Procedure SRand( Seed : RNG_IntType)
7594:  Function IRanGen : RNG_IntType
```

```
7595:  Function IRanGen31 : RNG_IntType
7596:  Function RanGen1 : Float
7597:  Function RanGen2 : Float
7598:  Function RanGen3 : Float
7599:  Function RanGen53 : Float
7600: end;
7601:
7602: //  Optimization by Simulated Annealing
7603: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7604: begin
7605:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7606:  Procedure SA_CreateLogFile( FileName : String)
7607:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7608: end;
7609:
7610: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7611: begin
7612:  Procedure InitUVAGbyString( KeyPhrase : string)
7613:  Procedure InitUVAG( Seed : RNG_IntType)
7614:  Function IRanUVAG : RNG_IntType
7615: end;
7616:
7617: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7618: begin
7619:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7620:  Procedure GA_CreateLogFile( LogFileName : String)
7621:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7622: end;
7623:
7624:  TVector', 'array of Float
7625: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7626: begin
7627:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7628:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7629: end;
7630:
7631: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7632: begin
7633:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7634:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7635: end;
7636:
7637: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7638: begin
7639:   FT_Result', 'Integer
7640:   //TDWordptr', '^DWord // will not work
7641:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7642:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7643:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7644:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7645:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7646:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7647:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7648:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7649:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7650:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7651:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7652:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7653:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7654:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7655:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7656:    yte; end
7657: end;
7658:
7659:
7660: //************************************ PaintFX*************************
7661: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7662: begin
7663:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7664:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7665:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7666:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7667:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7668:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7669:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7670:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7671:     Procedure Turn( Src, Dst : TBitmap)
7672:     Procedure TurnRight( Src, Dst : TBitmap)
7673:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7674:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7675:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7676:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7677:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7678:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7679:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7680:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7681:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7682:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7683:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
```

```
7684:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7685:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7686:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7687:     Procedure Emboss( var Bmp : TBitmap)
7688:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7689:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7690:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7691:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7692:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7693:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7694:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7695:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7696:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7697:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7698:     Procedure SemiOpaque( Src, Dst : TBitmap)
7699:     Procedure ShadowDownLeft( const Dst : TBitmap)
7700:     Procedure ShadowDownRight( const Dst : TBitmap)
7701:     Procedure ShadowUpLeft( const Dst : TBitmap)
7702:     Procedure ShadowUpRight( const Dst : TBitmap)
7703:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7704:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7705:     Procedure FlipRight( const Dst : TBitmap)
7706:     Procedure FlipDown( const Dst : TBitmap)
7707:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7708:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7709:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7710:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7711:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7712:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7713:     Procedure SmoothResize( var Src, Dst : TBitmap)
7714:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7715:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7716:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7717:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7718:     Procedure GrayScale( const Dst : TBitmap)
7719:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7720:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7721:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7722:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7723:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7724:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7725:     Procedure AntiAlias( const Dst : TBitmap)
7726:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7727:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7728:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7729:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7730:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7731:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7732:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7733:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7734:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7735:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7736:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7737:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7738:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7739:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7740:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7741:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7742:     Procedure Invert( Src : TBitmap)
7743:     Procedure MirrorRight( Src : TBitmap)
7744:     Procedure MirrorDown( Src : TBitmap)
7745:   end;
7746: end;
7747:
7748: (*----------------------------------------------------------------------------*)
7749: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7750: begin
7751:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7752:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7753:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7754:   SIRegister_TJvPaintFX(CL);
7755:  Function SplineFilter( Value : Single) : Single
7756:  Function BellFilter( Value : Single) : Single
7757:  Function TriangleFilter( Value : Single) : Single
7758:  Function BoxFilter( Value : Single) : Single
7759:  Function HermiteFilter( Value : Single) : Single
7760:  Function Lanczos3Filter( Value : Single) : Single
7761:  Function MitchellFilter( Value : Single) : Single
7762: end;
7763:
7764:
7765: (*----------------------------------------------------------------------------*)
7766: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7767: begin
7768:  'TeeMsg_DefaultFunctionName','String 'TeeFunction
7769:  TeeMsg_DefaultSeriesName','String 'Series
7770:  TeeMsg_DefaultToolName','String 'ChartTool
7771:  ChartComponentPalette','String 'TeeChart
7772:  TeeMaxLegendColumns',LongInt'( 2);
```

```
7773:  TeeDefaultLegendSymbolWidth','LongInt'( 20);
7774:  TeeTitleFootDistance,LongInt( 5);
7775:   SIRegister_TCustomChartWall(CL);
7776:   SIRegister_TChartWall(CL);
7777:   SIRegister_TChartLegendGradient(CL);
7778:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7779:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7780:   FindClass('TOBJECT'),'LegendException
7781:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7782:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7783:   FindClass('TOBJECT'),'TCustomChartLegend
7784:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7785:   TLegendSymbolPosition', '( spLeft, spRight )
7786:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7787:   TSymbolCalcHeight', 'Function  : Integer
7788:   SIRegister_TLegendSymbol(CL);
7789:   SIRegister_TTeeCustomShapePosition(CL);
7790:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7791:   SIRegister_TLegendTitle(CL);
7792:   SIRegister_TLegendItem(CL);
7793:   SIRegister_TLegendItems(CL);
7794:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7795:   FindClass('TOBJECT'),'TCustomChart
7796:   SIRegister_TCustomChartLegend(CL);
7797:   SIRegister_TChartLegend(CL);
7798:   SIRegister_TChartTitle(CL);
7799:   SIRegister_TChartFootTitle(CL);
7800:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7801:    +'eButton; Shift : TShiftState; X, Y : Integer)
7802:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7803:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7804:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7805:    +'TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7806:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7807:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7808:   TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7809:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7810:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7811:    +'toMax : Boolean; Min : Double; Max : Double; end
7812:   TAllAxisSavedScales', 'array of TAxisSavedScales
7813:   SIRegister_TChartBackWall(CL);
7814:   SIRegister_TChartRightWall(CL);
7815:   SIRegister_TChartBottomWall(CL);
7816:   SIRegister_TChartLeftWall(CL);
7817:   SIRegister_TChartWalls(CL);
7818:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7819:   SIRegister_TCustomChart(CL);
7820:   SIRegister_TChart(CL);
7821:   SIRegister_TTeeSeriesTypes(CL);
7822:   SIRegister_TTeeToolTypes(CL);
7823:   SIRegister_TTeeDragObject(CL);
7824:   SIRegister_TColorPalettes(CL);
7825:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7826:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7827:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7828:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7829:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7830:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7831:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7832:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7833:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7834:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7835:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7836:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7837:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7838:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7839:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7840:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7841:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7842:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7843:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7844:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7845:  Procedure PaintSeriesLegend(ASeries:TChartSeries;ACanvas:TCanvas;const
      R:TRect;ReferenceChart:TCustomChart);
7846:   SIRegister_TChartTheme(CL);
7847:   //TChartThemeClass', 'class of TChartTheme
7848:   //TCanvasClass', 'class of TCanvas3D
7849:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7850:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7851:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7852:  Procedure ShowMessageUser( const S : String)
7853:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7854:  Function HasLabels( ASeries : TChartSeries) : Boolean
7855:  Function HasColors( ASeries : TChartSeries) : Boolean
7856:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
```

```
7857:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7858: end;
7859:
7860:
7861: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7862: begin
7863:  //'TeeFormBorderStyle',' bsNone);
7864:   SIRegister_TMetafile(CL);
7865:  'TeeDefVerticalMargin','LongInt'( 4);
7866:  'TeeDefHorizMargin','LongInt'( 3);
7867:  'crTeeHand','LongInt'( TCursor ( 2020 ));
7868:  'TeeMsg_TeeHand','String 'crTeeHand
7869:  'TeeNormalPrintDetail','LongInt'( 0);
7870:  'TeeHighPrintDetail','LongInt'( - 100);
7871:  'TeeDefault_PrintMargin','LongInt'( 15);
7872:  'MaxDefaultColors','LongInt'( 19);
7873:  'TeeTabDelimiter','Char #9);
7874:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7875:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7876:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7877:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7878:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7879:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7880:   SIRegister_TCustomPanelNoCaption(CL);
7881:   FindClass('TOBJECT'),'TCustomTeePanel
7882:   SIRegister_TZoomPanning(CL);
7883:   SIRegister_TTeeEvent(CL);
7884:   //SIRegister_TTeeEventListeners(CL);
7885:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7886:   SIRegister_TTeeMouseEvent(CL);
7887:   SIRegister_TCustomTeePanel(CL);
7888:   //TChartGradient', 'TTeeGradient
7889:   //TChartGradientClass', 'class of TChartGradient
7890:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7891:   SIRegister_TTeeZoomPen(CL);
7892:   SIRegister_TTeeZoomBrush(CL);
7893:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7894:   SIRegister_TTeeZoom(CL);
7895:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7896:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7897:   SIRegister_TBackImage(CL);
7898:   SIRegister_TCustomTeePanelExtended(CL);
7899:   //TChartBrushClass', 'class of TChartBrush
7900:   SIRegister_TTeeCustomShapeBrushPen(CL);
7901:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7902:   TTextFormat', '( ttfNormal, ttfHtml )
7903:   SIRegister_TTeeCustomShape(CL);
7904:   SIRegister_TTeeShape(CL);
7905:   SIRegister_TTeeExportData(CL);
7906:  Function TeeStr( const Num : Integer) : String
7907:  Function DateTimeDefaultFormat( const AStep : Double) : String
7908:  Function TEEDaysInMonth( Year, Month : Word) : Word
7909:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7910:  Function NextDateTimeStep( const AStep : Double) : Double
7911:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7912:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7913:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7914:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7915:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7916:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7917:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7918:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7919:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7920:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7921:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7922:  Function DelphiToLocalFormat( const Format : String) : String
7923:  Function LocalToDelphiFormat( const Format : String) : String
7924:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
7925:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7926:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7927:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
7928:   TTeeSortSwap', 'Procedure ( a, b : Integer)
7929:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
7930:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7931:  Function TeeExtractField( St : String; Index : Integer) : String;
7932:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7933:  Function TeeNumFields( St : String) : Integer;
7934:  Function TeeNumFields1( const St, Separator : String) : Integer;
7935:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7936:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7937:  // TColorArray', 'array of TColor
7938:  Function GetDefaultColor( const Index : Integer) : TColor
7939:  Procedure SetDefaultColorPalette;
7940:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
7941:  'TeeCheckBoxSize','LongInt'( 11);
7942:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7943:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
7944:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
```

```
7945:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
7946:  Procedure TeeTranslateControl( AControl : TControl);
7947:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7948:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
7949:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
7950:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
7951:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
7952:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
7953:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
7954:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
7955:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
7956:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
7957:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
7958:  Procedure TeeSaveStringOption( const AKey, Value : String)
7959:  Function TeeDefaultXMLEncoding : String
7960:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
7961:   TeeWindowHandle', 'Integer
7962:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
7963:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
7964:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
7965: end;
7966:
7967:
7968: using mXBDEUtils
7969: ********************************************************************************
7970:  Procedure SetAlias( aAlias, aDirectory : String)
7971:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
7972:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
7973:  Procedure SetBDE( aPath, aNode, aValue : String)
7974:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
7975:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
7976:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
7977:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
7978:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
7979:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
7980:
7981:
7982: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
7983: begin
7984: AddClassN(FindClass('TOBJECT'),'EDateTime
7985: Function DatePart( const D : TDateTime) : Integer
7986: Function TimePart( const D : TDateTime) : Double
7987: Function Century( const D : TDateTime) : Word
7988: Function Year( const D : TDateTime) : Word
7989: Function Month( const D : TDateTime) : Word
7990: Function Day( const D : TDateTime) : Word
7991: Function Hour( const D : TDateTime) : Word
7992: Function Minute( const D : TDateTime) : Word
7993: Function Second( const D : TDateTime) : Word
7994: Function Millisecond( const D : TDateTime) : Word
7995: ('OneDay','Extended').setExtended( 1.0);
7996: ('OneHour','Extended').SetExtended( OneDay / 24);
7997: ('OneMinute','Extended').SetExtended( OneHour / 60);
7998: ('OneSecond','Extended').SetExtended( OneMinute / 60);
7999: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
8000: ('OneWeek','Extended').SetExtended( OneDay * 7);
8001: ('HoursPerDay','Extended').SetExtended( 24);
8002: ('MinutesPerHour','Extended').SetExtended( 60);
8003: ('SecondsPerMinute','Extended').SetExtended( 60);
8004: Procedure SetYear( var D : TDateTime; const Year : Word)
8005: Procedure SetMonth( var D : TDateTime; const Month : Word)
8006: Procedure SetDay( var D : TDateTime; const Day : Word)
8007: Procedure SetHour( var D : TDateTime; const Hour : Word)
8008: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8009: Procedure SetSecond( var D : TDateTime; const Second : Word)
8010: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8011: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8012: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8013: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8014: Function IsAM( const D : TDateTime) : Boolean
8015: Function IsPM( const D : TDateTime) : Boolean
8016: Function IsMidnight( const D : TDateTime) : Boolean
8017: Function IsNoon( const D : TDateTime) : Boolean
8018: Function IsSunday( const D : TDateTime) : Boolean
8019: Function IsMonday( const D : TDateTime) : Boolean
8020: Function IsTuesday( const D : TDateTime) : Boolean
8021: Function IsWedneday( const D : TDateTime) : Boolean
8022: Function IsThursday( const D : TDateTime) : Boolean
8023: Function IsFriday( const D : TDateTime) : Boolean
8024: Function IsSaturday( const D : TDateTime) : Boolean
8025: Function IsWeekend( const D : TDateTime) : Boolean
8026: Function Noon( const D : TDateTime) : TDateTime
8027: Function Midnight( const D : TDateTime) : TDateTime
8028: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8029: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8030: Function NextWorkday( const D : TDateTime) : TDateTime
8031: Function PreviousWorkday( const D : TDateTime) : TDateTime
8032: Function FirstDayOfYear( const D : TDateTime) : TDateTime
```

```
8033: Function LastDayOfYear( const D : TDateTime) : TDateTime
8034: Function EasterSunday( const Year : Word) : TDateTime
8035: Function GoodFriday( const Year : Word) : TDateTime
8036: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8037: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8038: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8039: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8040: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8041: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8042: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8043: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8044: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8045: Function DayOfYear( const D : TDateTime) : Integer
8046: Function DaysInMonth( const Ye, Mo : Word) : Integer
8047: Function DaysInMonth( const D : TDateTime) : Integer
8048: Function DaysInYear( const Ye : Word) : Integer
8049: Function DaysInYearDate( const D : TDateTime) : Integer
8050: Function WeekNumber( const D : TDateTime) : Integer
8051: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8052: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8053: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8054: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8055: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8056: Function DiffHours( const D1, D2 : TDateTime) : Integer
8057: Function DiffDays( const D1, D2 : TDateTime) : Integer
8058: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8059: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8060: Function DiffYears( const D1, D2 : TDateTime) : Integer
8061: Function GMTBias : Integer
8062: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8063: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8064: Function NowAsGMTTime : TDateTime
8065: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8066: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8067: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8068: Function DateTimeToANSI( const D : TDateTime) : Integer
8069: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8070: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8071: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8072: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8073: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8074: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8075: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8076: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8077: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8078: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8079: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8080: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8081: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8082: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8083: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8084: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8085: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8086: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8087: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8088: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8089: Function EnglishShortMonthA( const S : AnsiString) : Integer
8090: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8091: Function EnglishLongMonthA( const S : AnsiString) : Integer
8092: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8093: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8094: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8095: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8096: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8097: Function RFCMonthA( const S : AnsiString) : Word
8098: Function RFCMonthU( const S : UnicodeString) : Word
8099: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8100: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8101: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8102: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8103: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8104: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8105: Function NowAsRFCDateTimeA : AnsiString
8106: Function NowAsRFCDateTimeU : UnicodeString
8107: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8108: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8109: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8110: Function TimePeriodStr( const D : TDateTime) : AnsiString
8111: Procedure SelfTest
8112: end;
8113: //****************************************CFileUtils
8114: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8115: Function PathHasDriveLetter( const Path : String) : Boolean
8116: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8117: Function PathIsDriveLetter( const Path : String) : Boolean
8118: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8119: Function PathIsDriveRoot( const Path : String) : Boolean
8120: Function PathIsRootA( const Path : AnsiString) : Boolean
8121: Function PathIsRoot( const Path : String) : Boolean
```

```
8122: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8123: Function PathIsUNCPath( const Path : String) : Boolean
8124: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8125: Function PathIsAbsolute( const Path : String) : Boolean
8126: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8127: Function PathIsDirectory( const Path : String) : Boolean
8128: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8129: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8130: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8131: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8132: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8133: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8134: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8135: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8136: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8137: Function PathCanonical( const Path : String; const PathSep : Char) : String
8138: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8139: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8140: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8141: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8142: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8143: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8144: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8145: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8146: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8147: Function FileNameValid( const FileName : String) : String
8148: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8149: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8150: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8151: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8152: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8153: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8154: Procedure CCopyFile( const FileName, DestName : String)
8155: Procedure CMoveFile( const FileName, DestName : String)
8156: Function CDeleteFiles( const FileMask : String) : Boolean
8157: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8158: Procedure FileCloseEx( const FileHandle : TFileHandle)
8159: Function FileExistsA( const FileName : AnsiString) : Boolean
8160:  Function CFileExists( const FileName : String) : Boolean
8161:  Function CFileGetSize( const FileName : String) : Int64
8162: Function FileGetDateTime( const FileName : String) : TDateTime
8163: Function FileGetDateTime2( const FileName : String) : TDateTime
8164: Function FileIsReadOnly( const FileName : String) : Boolean
8165: Procedure FileDeleteEx( const FileName : String)
8166: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8167: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
      : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8168: Function DirectoryEntryExists( const Name : String) : Boolean
8169: Function DirectoryEntrySize( const Name : String) : Int64
8170:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8171: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8172: Procedure CDirectoryCreate( const DirectoryName : String)
8173: Function GetFirstFileNameMatching( const FileMask : String) : String
8174: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8175: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8176: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8177:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8178:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8179: Function DriveIsValid( const Drive : Char) : Boolean
8180: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8181: Function DriveFreeSpace( const Path : AnsiString) : Int64
8182:
8183: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8184: begin
8185:  AddClassN(FindClass('TOBJECT'),'ETimers
8186:  Const('TickFrequency','LongInt'( 1000);Function GetTick : LongWord
8187: Function TickDelta( const D1, D2 : LongWord) : Integer
8188: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8189:   AddTypeS('THPTimer', 'Int64
8190: Procedure StartTimer( var Timer : THPTimer)
8191: Procedure StopTimer( var Timer : THPTimer)
8192: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8193: Procedure InitStoppedTimer( var Timer : THPTimer)
8194: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8195: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8196: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8197: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8198: Function GetHighPrecisionFrequency : Int64
8199: Function GetHighPrecisionTimerOverhead : Int64
8200: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8201: Procedure SelfTestCTimer
8202: end;
8203:
8204: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8205: begin
8206:  Function RandomSeed : LongWord
8207:  Procedure AddEntropy( const Value : LongWord)
8208:  Function RandomUniform : LongWord;
8209:  Function RandomUniform1( const N : Integer) : Integer;
```

```
8210:  Function RandomBoolean : Boolean
8211:  Function RandomByte : Byte
8212:  Function RandomByteNonZero : Byte
8213:  Function RandomWord : Word
8214:  Function RandomInt64 : Int64;
8215:  Function RandomInt641( const N : Int64) : Int64;
8216:  Function RandomHex( const Digits : Integer) : String
8217:  Function RandomFloat : Extended
8218:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8219:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8220:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8221:  Function mwcRandomLongWord : LongWord
8222:  Function urnRandomLongWord : LongWord
8223:  Function moaRandomFloat : Extended
8224:  Function mwcRandomFloat : Extended
8225:  Function RandomNormalF : Extended
8226:  Procedure SelfTestCRandom
8227: end;
8228:
8229: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8230: begin
8231:  // PIntArray', '^TIntArray // will not work
8232:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8233:  TConvertTabsProcEx,function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8234:  Function synMax( x, y : integer) : integer
8235:  Function synMin( x, y : integer) : integer
8236:  Function synMinMax( x, mi, ma : integer) : integer
8237:  Procedure synSwapInt( var l, r : integer)
8238:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8239:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8240:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8241:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8242:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8243:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8244:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8245:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8246:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8247:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8248:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8249:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8250:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8251:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8252:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8253:  ('C3_NONSPACING','LongInt'( 1);
8254:  'C3_DIACRITIC','LongInt'( 2);
8255:  'C3_VOWELMARK','LongInt'( 4);
8256:  ('C3_SYMBOL','LongInt'( 8);
8257:  ('C3_KATAKANA','LongWord( $0010);
8258:  ('C3_HIRAGANA','LongWord( $0020);
8259:  ('C3_HALFWIDTH','LongWord( $0040);
8260:  ('C3_FULLWIDTH','LongWord( $0080);
8261:  ('C3_IDEOGRAPH','LongWord( $0100);
8262:  ('C3_KASHIDA','LongWord( $0200);
8263:  ('C3_LEXICAL','LongWord( $0400);
8264:  ('C3_ALPHA','LongWord( $8000);
8265:  ('C3_NOTAPPLICABLE','LongInt'( 0);
8266:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8267:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8268:  Function synIsStringType( Value : Word) : TStringType
8269:  Function synGetEOL( Line : PChar) : PChar
8270:  Function synEncodeString( s : string) : string
8271:  Function synDecodeString( s : string) : string
8272:  Procedure synFreeAndNil( var Obj: TObject)
8273:  Procedure synAssert( Expr : Boolean)
8274:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8275:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8276:   TReplaceFlags', 'set of TReplaceFlag )
8277:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8278:  Function synGetRValue( RGBValue : TColor) : byte
8279:  Function synGetGValue( RGBValue : TColor) : byte
8280:  Function synGetBValue( RGBValue : TColor) : byte
8281:  Function synRGB( r, g, b : Byte) : Cardinal
8282: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8283:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8284:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8285:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8286:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8287: end;
8288:
8289:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8290:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8291:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8292:
8293: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8294: begin
8295:  Function STimeZoneBias : integer
8296:  Function TimeZone : string
```

```
8297:  Function Rfc822DateTime( t : TDateTime) : string
8298:  Function CDateTime( t : TDateTime) : string
8299:  Function SimpleDateTime( t : TDateTime) : string
8300:  Function AnsiCDateTime( t : TDateTime) : string
8301:  Function GetMonthNumber( Value : String) : integer
8302:  Function GetTimeFromStr( Value : string) : TDateTime
8303:  Function GetDateMDYFromStr( Value : string) : TDateTime
8304:  Function DecodeRfcDateTime( Value : string) : TDateTime
8305:  Function GetUTTime : TDateTime
8306:  Function SetUTTime( Newdt : TDateTime) : Boolean
8307:  Function SGetTick : LongWord
8308:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8309:  Function CodeInt( Value : Word) : Ansistring
8310:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8311:  Function CodeLongInt( Value : LongInt) : Ansistring
8312:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8313:  Function DumpStr( const Buffer : Ansistring) : string
8314:  Function DumpExStr( const Buffer : Ansistring) : string
8315:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8316:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8317:  Function TrimSPLeft( const S : string) : string
8318:  Function TrimSPRight( const S : string) : string
8319:  Function TrimSP( const S : string) : string
8320:  Function SeparateLeft( const Value, Delimiter : string) : string
8321:  Function SeparateRight( const Value, Delimiter : string) : string
8322:  Function SGetParameter( const Value, Parameter : string) : string
8323:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8324:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8325:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8326:  Function GetEmailAddr( const Value : string) : string
8327:  Function GetEmailDesc( Value : string) : string
8328:  Function CStrToHex( const Value : Ansistring) : string
8329:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8330:  Function CBinToInt( const Value : string) : Integer
8331:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8332:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8333:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8334:  Function CRPos( const Sub, Value : String) : Integer
8335:  Function FetchBin( var Value : string; const Delimiter : string) : string
8336:  Function CFetch( var Value : string; const Delimiter : string) : string
8337:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8338:  Function IsBinaryString( const Value : AnsiString) : Boolean
8339:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8340:  Procedure StringsTrim( const value : TStrings)
8341:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8342:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8343:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8344:  Function CCountOfChar( const Value : string; aChr : char) : integer
8345:  Function UnquoteStr( const Value : string; Quote : Char) : string
8346:  Function QuoteStr( const Value : string; Quote : Char) : string
8347:  Procedure HeadersToList( const Value : TStrings)
8348:  Procedure ListToHeaders( const Value : TStrings)
8349:  Function SwapBytes( Value : integer) : integer
8350:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8351:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8352:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8353:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8354:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8355:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8356: end;
8357:
8358: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8359: begin
8360:  ('CrcBufSize','LongInt'( 2048);
8361:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8362:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8363:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8364:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8365:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8366:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8367:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8368:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8369:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8370:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8371:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8372:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8373:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8374:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8375:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8376: end;
8377:
8378: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8379: begin
8380:  function CreateOleObject(const ClassName: String): IDispatch;
8381:  function GetActiveOleObject(const ClassName: String): IDispatch;
8382:  function ProgIDToClassID(const ProgID: string): TGUID;
8383:  function ClassIDToProgID(const ClassID: TGUID): string;
8384:  function CreateClassID: string;
8385:  function CreateGUIDString: string;
```

```
8386:  function CreateGUIDID: string;
8387:  procedure OleError(ErrorCode: longint)
8388:  procedure OleCheck(Result: HResult);
8389: end;
8390:
8391:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8392:  Function xGetActiveOleObject( const ClassName : string) : Variant
8393:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8394:  Function DllCanUnloadNow : HResult
8395:  Function DllRegisterServer : HResult
8396:  Function DllUnregisterServer : HResult
8397:  Function VarFromInterface( Unknown : IUnknown) : Variant
8398:  Function VarToInterface( const V : Variant) : IDispatch
8399:  Function VarToAutoObject( const V : Variant) : TAutoObject
8400:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8401:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8402:  Procedure OleError( ErrorCode : HResult)
8403:  Procedure OleCheck( Result : HResult)
8404:  Function StringToClassID( const S : string) : TCLSID
8405:  Function ClassIDToString( const ClassID : TCLSID) : string
8406:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8407:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8408:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8409:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8410:  Function xGUIDToString( const ClassID : TGUID) : string
8411:  Function xStringToGUID( const S : string) : TGUID
8412:  Function xGetModuleName( Module : HMODULE) : string
8413:  Function xAcquireExceptionObject : TObject
8414:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8415:  Function xUtf8Encode( const WS : WideString) : UTF8String
8416:  Function xUtf8Decode( const S : UTF8String) : WideString
8417:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8418:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8419:  Function XRTLHandleCOMException : HResult
8420:  Procedure XRTLCheckArgument( Flag : Boolean)
8421:  //Procedure XRTLCheckOutArgument( out Arg)
8422:  Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8423:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8424:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8425:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8426:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8427:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8428: function   XRTLDefaultCategoryManager: IUnknown;
8429: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8430: // ICatRegister helper functions
8431: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8432:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8433:                                        const CategoryManager: IUnknown = nil): HResult;
8434: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8435:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8436:                                        const CategoryManager: IUnknown = nil): HResult;
8437: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8438:                                        const CategoryManager: IUnknown = nil): HResult;
8439: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8440:                                          const CategoryManager: IUnknown = nil): HResult;
8441: // ICatInformation helper functions
8442: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8443:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8444:                                       const CategoryManager: IUnknown = nil): HResult;
8445: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8446:                                const CategoryManager: IUnknown = nil): HResult;
8447: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8448:                                     const CategoryManager: IUnknown = nil): HResult;
8449: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8450:                                      const CategoryManager: IUnknown = nil): HResult;
8451: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8452:                    const ADelete: Boolean = True): WideString;
8453: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8454: Function XRTLGetVariantAsString( const Value : Variant) : string
8455: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8456: Function XRTLGetTimeZones : TXRTLTimeZones
8457: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8458: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8459: Function GMTNow : TDateTime
8460: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8461: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8462: Procedure XRTLNotImplemented
8463: Procedure XRTLRaiseError( E : Exception)
8464: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8465:
8466:
8467: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8468: begin
8469:   SIRegister_IXRTLValue(CL);
8470:   SIRegister_TXRTLValue(CL);
8471:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
```

```
8472:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8473: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8474: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8475: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8476: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8477: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8478: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8479: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8480: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8481: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8482: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8483: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8484: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8485: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8486: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8487: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8488: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8489: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8490: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8491: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8492: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8493: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8494: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8495: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8496: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8497: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8498: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8499: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8500:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8501: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8502: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8503: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8504: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8505: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8506: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8507: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8508: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8509: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8510: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8511: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8512: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8513: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8514: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8515: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8516: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8517: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8518: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8519: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8520: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8521: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8522: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8523: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8524: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8525: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8526: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8527: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8528: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8529: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8530: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8531: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8532: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8533: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8534: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8535: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8536: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8537: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8538: end;
8539:
8540: //*****************************unit uPSI_GR32;****************************************
8541:
8542:  Function Color32( WinColor : TColor) : TColor32;
8543:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8544:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8545:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8546:  Function WinColor( Color32 : TColor32) : TColor
8547:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8548:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8549:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8550:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8551:  Function RedComponent( Color32 : TColor32) : Integer
8552:  Function GreenComponent( Color32 : TColor32) : Integer
8553:  Function BlueComponent( Color32 : TColor32) : Integer
8554:  Function AlphaComponent( Color32 : TColor32) : Integer
8555:  Function Intensity( Color32 : TColor32) : Integer
8556:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8557:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8558:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8559:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
```

```
8560:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8561:  Function WinPalette( const P : TPalette32) : HPALETTE
8562:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8563:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8564:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8565:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8566:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
8567:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8568:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8569:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8570:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8571:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8572:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8573:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8574:  Function FixedRect1( const ARect : TRect) : TRect;
8575:  Function FixedRect2( const FR : TFloatRect) : TRect;
8576:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8577:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8578:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8579:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8580:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8581:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8582:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8583:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8584:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8585:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8586:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8587:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8588:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8589:  Function IsRectEmpty( const R : TRect) : Boolean;
8590:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8591:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8592:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8593:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8594:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8595:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8596:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8597:  Function MessageBeep( uType : UINT) : BOOL
8598:  Function ShowCursor( bShow : BOOL) : Integer
8599:  Function SetCursorPos( X, Y : Integer) : BOOL
8600:  Function SetCursor( hCursor : HICON) : HCURSOR
8601:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8602:  //Function ClipCursor( lpRect : PRect) : BOOL
8603:  Function GetClipCursor( var lpRect : TRect) : BOOL
8604:  Function GetCursor : HCURSOR
8605:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8606:  Function GetCaretBlinkTime : UINT
8607:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8608:  Function DestroyCaret : BOOL
8609:  Function HideCaret( hWnd : HWND) : BOOL
8610:  Function ShowCaret( hWnd : HWND) : BOOL
8611:  Function SetCaretPos( X, Y : Integer) : BOOL
8612:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8613:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8614:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8615:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8616:  Function WindowFromPoint( Point : TPoint) : HWND
8617:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8618:
8619:
8620: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8621: begin
8622:  Function FixedFloor( A : TFixed) : Integer
8623:  Function FixedCeil( A : TFixed) : Integer
8624:  Function FixedMul( A, B : TFixed) : TFixed
8625:  Function FixedDiv( A, B : TFixed) : TFixed
8626:  Function OneOver( Value : TFixed) : TFixed
8627:  Function FixedRound( A : TFixed) : Integer
8628:  Function FixedSqr( Value : TFixed) : TFixed
8629:  Function FixedSqrtLP( Value : TFixed) : TFixed
8630:  Function FixedSqrtHP( Value : TFixed) : TFixed
8631:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8632:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8633:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8634:  Function GRHypot( const X, Y : TFloat) : TFloat;
8635:  Function Hypot1( const X, Y : Integer) : Integer;
8636:  Function FastSqrt( const Value : TFloat) : TFloat
8637:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8638:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8639:  Function FastInvSqrt( const Value : Single) : Single;
8640:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8641:  Function GRIsPowerOf2( Value : Integer) : Boolean
8642:  Function PrevPowerOf2( Value : Integer) : Integer
8643:  Function NextPowerOf2( Value : Integer) : Integer
8644:  Function Average( A, B : Integer) : Integer
8645:  Function GRSign( Value : Integer) : Integer
8646:  Function FloatMod( x, y : Double) : Double
8647: end;
8648:
```

```
8649: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8650: begin
8651:  Function Clamp( const Value : Integer) : Integer;
8652:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8653:  Function StackAlloc( Size : Integer) : Pointer
8654:  Procedure StackFree( P : Pointer)
8655:  Procedure Swap( var A, B : Pointer);
8656:  Procedure Swap1( var A, B : Integer);
8657:  Procedure Swap2( var A, B : TFixed);
8658:  Procedure Swap3( var A, B : TColor32);
8659:  Procedure TestSwap( var A, B : Integer);
8660:  Procedure TestSwap1( var A, B : TFixed);
8661:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8662:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8663:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8664:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8665:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8666:  Function GRMin( const A, B, C : Integer) : Integer;
8667:  Function GRMax( const A, B, C : Integer) : Integer;
8668:  Function Clamp( Value, Max : Integer) : Integer;
8669:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8670:  Function Wrap( Value, Max : Integer) : Integer;
8671:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8672:  Function Wrap3( Value, Max : Single) : Single;;
8673:  Function WrapPow2( Value, Max : Integer) : Integer;
8674:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8675:  Function Mirror( Value, Max : Integer) : Integer;
8676:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8677:  Function MirrorPow2( Value, Max : Integer) : Integer;
8678:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8679:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8680:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8681:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8682:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8683:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8684:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8685:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8686:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8687:  Function Div255( Value : Cardinal) : Cardinal
8688:  Function SAR_4( Value : Integer) : Integer
8689:  Function SAR_8( Value : Integer) : Integer
8690:  Function SAR_9( Value : Integer) : Integer
8691:  Function SAR_11( Value : Integer) : Integer
8692:  Function SAR_12( Value : Integer) : Integer
8693:  Function SAR_13( Value : Integer) : Integer
8694:  Function SAR_14( Value : Integer) : Integer
8695:  Function SAR_15( Value : Integer) : Integer
8696:  Function SAR_16( Value : Integer) : Integer
8697:  Function ColorSwap( WinColor : TColor) : TColor32
8698: end;
8699:
8700: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8701: begin
8702:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8703:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8704:  Procedure CopyComponents1(Dst:TCustBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8705:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8706:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8707:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8708:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8709:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8710:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8711:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8712:  Function CreateBitmask( Components : TColor32Components) : TColor32
8713:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8714:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8715:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8716: end;
8717:
8718:
8719: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8720: begin
8721:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8722:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8723:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8724:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8725:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8726:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8727:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8728:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8729:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8730:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8731:    //+'tedRangeBuffer; MoreData : Boolean; end
8732:  Function NtfsSetSparse( const FileName : string) : Boolean
8733:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8734:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
```

```
8735:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8736:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8737:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8738:  Function NtfsGetSparse( const FileName : string) : Boolean
8739:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
8740:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8741:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8742:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8743:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8744:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8745:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8746:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8747:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8748:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8749:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8750:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8751:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8752:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8753:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8754:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8755:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8756:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8757:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8758:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8759:   AddTypeS('TStreamIds', 'set of TStreamId
8760:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8761:    +': ___Pointer; StreamIds : TStreamIds; end
8762:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8763:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8764:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8765:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8766:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8767:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8768:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8769:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8770:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8771:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8772:  Function JclAppInstances : TJclAppInstances;
8773:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8774:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8775:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8776:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8777:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8778:
8779:
8780: (*----------------------------------------------------------------------------*)
8781: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8782: begin
8783:   FindClass('TOBJECT'),'EJclGraphicsError
8784:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8785:   TDynPointArray', 'array of TPoint
8786:   TDynDynPointArrayArray', 'array of TDynPointArray
8787:   TPointF', 'record X : Single; Y : Single; end
8788:   TDynPointArrayF', 'array of TPointF
8789:   TDrawMode2', '( dmOpaque, dmBlend )
8790:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8791:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8792:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8793:   TMatrix3d', 'record array[0..2,0..2] of extended end
8794:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8795:   TScanLine', 'array of Integer
8796:   TScanLines', 'array of TScanLine
8797:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8798:   TGradientDirection', '( gdVertical, gdHorizontal )
8799:   TPolyFillMode', '( fmAlternate, fmWinding )
8800:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8801:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8802:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8803:   SIRegister_TJclDesktopCanvas(CL);
8804:   FindClass('TOBJECT'),'TJclRegion
8805:   SIRegister_TJclRegionInfo(CL);
8806:   SIRegister_TJclRegion(CL);
8807:   SIRegister_TJclThreadPersistent(CL);
8808:   SIRegister_TJclCustomMap(CL);
8809:   SIRegister_TJclBitmap32(CL);
8810:   SIRegister_TJclByteMap(CL);
8811:   SIRegister_TJclTransformation(CL);
8812:   SIRegister_TJclLinearTransformation(CL);
8813:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8814:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8815:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8816:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8817:  Procedure BitmapToJPeg( const FileName : string)
8818:  Procedure JPegToBitmap( const FileName : string)
8819:  Function ExtractIconCount( const FileName : string) : Integer
```

```
8820:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8821:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8822:  Procedure
      BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8823:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8824:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8825:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8826:  Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
      TGradientDirection) : Boolean;
8827:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8828:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8829:  Procedure ScreenShot1( bm : TBitmap);
8830:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8831:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8832:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8833:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8834:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8835:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8836:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8837:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8838:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8839:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8840:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8841:  Procedure Invert( Dst, Src : TJclBitmap32)
8842:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8843:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8844:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8845:  Procedure SetGamma( Gamma : Single)
8846: end;
8847:
8848: (*----------------------------------------------------------------------------*)
8849: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8850: begin
8851:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8852:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8853:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8854:  Function LockedDec( var Target : Integer) : Integer
8855:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8856:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8857:  Function LockedExchangeDec( var Target : Integer) : Integer
8858:  Function LockedExchangeInc( var Target : Integer) : Integer
8859:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8860:  Function LockedInc( var Target : Integer) : Integer
8861:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8862:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8863:   SIRegister_TJclDispatcherObject(CL);
8864:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8865:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8866:   SIRegister_TJclCriticalSection(CL);
8867:   SIRegister_TJclCriticalSectionEx(CL);
8868:   SIRegister_TJclEvent(CL);
8869:   SIRegister_TJclWaitableTimer(CL);
8870:   SIRegister_TJclSemaphore(CL);
8871:   SIRegister_TJclMutex(CL);
8872:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8873:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8874:   SIRegister_TJclOptex(CL);
8875:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8876:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8877:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8878:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8879:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8880:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8881:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8882:   PMeteredSection', '^TMeteredSection // will not work
8883:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8884:   SIRegister_TJclMeteredSection(CL);
8885:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8886:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8887:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8888:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8889:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8890:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8891:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8892:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8893:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8894:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8895:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8896:   FindClass('TOBJECT'),'EJclCriticalSectionError
8897:   FindClass('TOBJECT'),'EJclEventError
8898:   FindClass('TOBJECT'),'EJclWaitableTimerError
8899:   FindClass('TOBJECT'),'EJclSemaphoreError
8900:   FindClass('TOBJECT'),'EJclMutexError
8901:   FindClass('TOBJECT'),'EJclMeteredSectionError
8902: end;
```

```
8903:
8904:
8905: //*************************unit uPSI_mORMotReport;
8906: Procedure SetCurrentPrinterAsDefault
8907: Function CurrentPrinterName : string
8908: Function mCurrentPrinterPaperSize : string
8909: Procedure UseDefaultPrinter
8910:
8911: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
8912: begin
8913:   with FindClass('TOBJECT'), 'TStream') do begin
8914:     IsAbstract := True;
8915:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8916:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8917:     function Read(Buffer:String;Count:LongInt):LongInt
8918:     function Write(Buffer:String;Count:LongInt):LongInt
8919:     function ReadString(Buffer:String;Count:LongInt):LongInt    //FileStream
8920:     function WriteString(Buffer:String;Count:LongInt):LongInt
8921:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8922:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
8923:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8924:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt');
8925:
8926:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8927:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8928:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8929:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8930:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8931:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8932:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8933:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8934:
8935:     function Seek(Offset:LongInt;Origin:Word):LongInt
8936:     procedure ReadBuffer(Buffer:String;Count:LongInt)
8937:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8938:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
8939:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)');
8940:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)');
8941:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)');
8942:
8943:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8944:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8945:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8946:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8947:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8948:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8949:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8950:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8951:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8952:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8953:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
8954:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
8955:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
8956:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
8957:
8958:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
8959:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
8960:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)');
8961:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)');
8962:     //READBUFFERAC
8963:     function InstanceSize: Longint
8964:     Procedure FixupResourceHeader( FixupInfo : Integer)
8965:     Procedure ReadResHeader
8966:
8967:     {$IFDEF DELPHI4UP}
8968:     function CopyFrom(Source:TStream;Count:Int64):LongInt
8969:     {$ELSE}
8970:     function CopyFrom(Source:TStream;Count:Integer):LongInt
8971:     {$ENDIF}
8972:     RegisterProperty('Position', 'LongInt', iptrw);
8973:     RegisterProperty('Size', 'LongInt', iptrw);
8974:   end;
8975: end;
8976:
8977:
8978: { ***************************************************************
8979:   Unit DMATH - Interface for DMATH.DLL
8980:    *************************************************************** }
8981: // see more docs/dmath_manual.pdf
8982:
8983: Function InitEval : Integer
8984: Procedure SetVariable( VarName : Char; Value : Float)
8985: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
8986: Function Eval( ExpressionString : String) : Float
8987:
8988: unit dmath; //types are in built, others are external in DLL
8989: interface
8990: {$IFDEF DELPHI}
8991: uses
```

```
8992:   StdCtrls, Graphics;
8993: {$ENDIF}
8994: { ---------------------------------------------------------------
8995:   Types and constants
8996:   --------------------------------------------------------------- }
8997: {$i types.inc}
8998: { ---------------------------------------------------------------
8999:   Error handling
9000:   --------------------------------------------------------------- }
9001: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9002: { Sets the error code }
9003: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9004: { Sets error code and default function value }
9005: function MathErr : Integer; external 'dmath';
9006: { Returns the error code }
9007: { ---------------------------------------------------------------
9008:   Dynamic arrays
9009:   --------------------------------------------------------------- }
9010: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9011: { Sets the auto-initialization of arrays }
9012: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9013: { Creates floating point vector V[0..Ub] }
9014: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9015: { Creates integer vector V[0..Ub] }
9016: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9017: { Creates complex vector V[0..Ub] }
9018: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9019: { Creates boolean vector V[0..Ub] }
9020: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9021: { Creates string vector V[0..Ub] }
9022: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9023: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9024: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9025: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9026: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9027: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9028: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9029: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9030: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9031: { Creates string matrix A[0..Ub1, 0..Ub2] }
9032: { ---------------------------------------------------------------
9033:   Minimum, maximum, sign and exchange
9034:   --------------------------------------------------------------- }
9035: function FMin(X, Y : Float) : Float; external 'dmath';
9036: { Minimum of 2 reals }
9037: function FMax(X, Y : Float) : Float; external 'dmath';
9038: { Maximum of 2 reals }
9039: function IMin(X, Y : Integer) : Integer; external 'dmath';
9040: { Minimum of 2 integers }
9041: function IMax(X, Y : Integer) : Integer; external 'dmath';
9042: { Maximum of 2 integers }
9043: function Sgn(X : Float) : Integer; external 'dmath';
9044: { Sign (returns 1 if X = 0) }
9045: function Sgn0(X : Float) : Integer; external 'dmath';
9046: { Sign (returns 0 if X = 0) }
9047: function DSgn(A, B : Float) : Float; external 'dmath';
9048: { Sgn(B) * |A| }
9049: procedure FSwap(var X, Y : Float); external 'dmath';
9050: { Exchange 2 reals }
9051: procedure ISwap(var X, Y : Integer); external 'dmath';
9052: { Exchange 2 integers }
9053: { ---------------------------------------------------------------
9054:   Rounding functions
9055:   --------------------------------------------------------------- }
9056: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9057: { Rounds X to N decimal places }
9058: function Ceil(X : Float) : Integer; external 'dmath';
9059: { Ceiling function }
9060: function Floor(X : Float) : Integer; external 'dmath';
9061: { Floor function }
9062: { ---------------------------------------------------------------
9063:   Logarithms, exponentials and power
9064:   --------------------------------------------------------------- }
9065: function Expo(X : Float) : Float; external 'dmath';
9066: { Exponential }
9067: function Exp2(X : Float) : Float; external 'dmath';
9068: { 2^X }
9069: function Exp10(X : Float) : Float; external 'dmath';
9070: { 10^X }
9071: function Log(X : Float) : Float; external 'dmath';
9072: { Natural log }
9073: function Log2(X : Float) : Float; external 'dmath';
9074: { Log, base 2 }
9075: function Log10(X : Float) : Float; external 'dmath';
9076: { Decimal log }
9077: function LogA(X, A : Float) : Float; external 'dmath';
9078: { Log, base A }
9079: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9080: { X^N }
```

```
9081: function Power(X, Y : Float) : Float; external 'dmath';
9082: { X^Y, X >= 0 }
9083: { ---------------------------------------------------------------
9084:   Trigonometric functions
9085:   --------------------------------------------------------------- }
9086: function Pythag(X, Y : Float) : Float; external 'dmath';
9087: { Sqrt(X^2 + Y^2) }
9088: function FixAngle(Theta : Float) : Float; external 'dmath';
9089: { Set Theta in -Pi..Pi }
9090: function Tan(X : Float) : Float; external 'dmath';
9091: { Tangent }
9092: function ArcSin(X : Float) : Float; external 'dmath';
9093: { Arc sinus }
9094: function ArcCos(X : Float) : Float; external 'dmath';
9095: { Arc cosinus }
9096: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9097: { Angle (Ox, OM) with M(X,Y) }
9098: { ---------------------------------------------------------------
9099:   Hyperbolic functions
9100:   --------------------------------------------------------------- }
9101: function Sinh(X : Float) : Float; external 'dmath';
9102: { Hyperbolic sine }
9103: function Cosh(X : Float) : Float; external 'dmath';
9104: { Hyperbolic cosine }
9105: function Tanh(X : Float) : Float; external 'dmath';
9106: { Hyperbolic tangent }
9107: function ArcSinh(X : Float) : Float; external 'dmath';
9108: { Inverse hyperbolic sine }
9109: function ArcCosh(X : Float) : Float; external 'dmath';
9110: { Inverse hyperbolic cosine }
9111: function ArcTanh(X : Float) : Float; external 'dmath';
9112: { Inverse hyperbolic tangent }
9113: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9114: { Sinh & Cosh }
9115: { ---------------------------------------------------------------
9116:   Gamma function and related functions
9117:   --------------------------------------------------------------- }
9118: function Fact(N : Integer) : Float; external 'dmath';
9119: { Factorial }
9120: function SgnGamma(X : Float) : Integer; external 'dmath';
9121: { Sign of Gamma function }
9122: function Gamma(X : Float) : Float; external 'dmath';
9123: { Gamma function }
9124: function LnGamma(X : Float) : Float; external 'dmath';
9125: { Logarithm of Gamma function }
9126: function Stirling(X : Float) : Float; external 'dmath';
9127: { Stirling's formula for the Gamma function }
9128: function StirLog(X : Float) : Float; external 'dmath';
9129: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9130: function DiGamma(X : Float ) : Float; external 'dmath';
9131: { Digamma function }
9132: function TriGamma(X : Float ) : Float; external 'dmath';
9133: { Trigamma function }
9134: function IGamma(A, X : Float) : Float; external 'dmath';
9135: { Incomplete Gamma function}
9136: function JGamma(A, X : Float) : Float; external 'dmath';
9137: { Complement of incomplete Gamma function }
9138: function InvGamma(A, P : Float) : Float; external 'dmath';
9139: { Inverse of incomplete Gamma function }
9140: function Erf(X : Float) : Float; external 'dmath';
9141: { Error function }
9142: function Erfc(X : Float) : Float; external 'dmath';
9143: { Complement of error function }
9144: { ---------------------------------------------------------------
9145:   Beta function and related functions
9146:   --------------------------------------------------------------- }
9147: function Beta(X, Y : Float) : Float; external 'dmath';
9148: { Beta function }
9149: function IBeta(A, B, X : Float) : Float; external 'dmath';
9150: { Incomplete Beta function }
9151: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9152: { Inverse of incomplete Beta function }
9153: { ---------------------------------------------------------------
9154:   Lambert's function
9155:   --------------------------------------------------------------- }
9156: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9157:  ---------------------------------------------------------------
9158:   Binomial distribution
9159:   --------------------------------------------------------------- }
9160: function Binomial(N, K : Integer) : Float; external 'dmath';
9161: { Binomial coefficient C(N,K) }
9162: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9163: { Probability of binomial distribution }
9164: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9165: { Cumulative probability for binomial distrib. }
9166: { ---------------------------------------------------------------
9167:   Poisson distribution
9168:   --------------------------------------------------------------- }
9169: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
```

```
9170: { Probability of Poisson distribution }
9171: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9172: { Cumulative probability for Poisson distrib. }
9173: { ------------------------------------------------------------
9174:   Exponential distribution
9175:   ------------------------------------------------------------ }
9176: function DExpo(A, X : Float) : Float; external 'dmath';
9177: { Density of exponential distribution with parameter A }
9178: function FExpo(A, X : Float) : Float; external 'dmath';
9179: { Cumulative probability function for exponential dist. with parameter A }
9180: { ------------------------------------------------------------
9181:   Standard normal distribution
9182:   ------------------------------------------------------------ }
9183: function DNorm(X : Float) : Float; external 'dmath';
9184: { Density of standard normal distribution }
9185: function FNorm(X : Float) : Float; external 'dmath';
9186: { Cumulative probability for standard normal distrib. }
9187: function PNorm(X : Float) : Float; external 'dmath';
9188: { Prob(|U| > X) for standard normal distrib. }
9189: function InvNorm(P : Float) : Float; external 'dmath';
9190: { Inverse of standard normal distribution }
9191: { ------------------------------------------------------------
9192:   Student's distribution
9193:   ------------------------------------------------------------ }
9194: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9195: { Density of Student distribution with Nu d.o.f. }
9196: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9197: { Cumulative probability for Student distrib. with Nu d.o.f. }
9198: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9199: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9200: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9201: { Inverse of Student's t-distribution function }
9202: { ------------------------------------------------------------
9203:   Khi-2 distribution
9204:   ------------------------------------------------------------ }
9205: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9206: { Density of Khi-2 distribution with Nu d.o.f. }
9207: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9208: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9209: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9210: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9211: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9212: { Inverse of Khi-2 distribution function }
9213: { ------------------------------------------------------------
9214:   Fisher-Snedecor distribution
9215:   ------------------------------------------------------------ }
9216: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9217: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9218: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9219: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9220: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9221: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9222: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9223: { Inverse of Snedecor's F-distribution function }
9224: { ------------------------------------------------------------
9225:   Beta distribution
9226:   ------------------------------------------------------------ }
9227: function DBeta(A, B, X : Float) : Float; external 'dmath';
9228: { Density of Beta distribution with parameters A and B }
9229: function FBeta(A, B, X : Float) : Float; external 'dmath';
9230: { Cumulative probability for Beta distrib. with param. A and B }
9231: { ------------------------------------------------------------
9232:   Gamma distribution
9233:   ------------------------------------------------------------ }
9234: function DGamma(A, B, X : Float) : Float; external 'dmath';
9235: { Density of Gamma distribution with parameters A and B }
9236: function FGamma(A, B, X : Float) : Float; external 'dmath';
9237: { Cumulative probability for Gamma distrib. with param. A and B }
9238: { ------------------------------------------------------------
9239:   Expression evaluation
9240:   ------------------------------------------------------------ }
9241: function InitEval : Integer; external 'dmath';
9242: { Initializes built-in functions and returns their number }
9243: function Eval(ExpressionString : String) : Float; external 'dmath';
9244: { Evaluates an expression at run-time }
9245: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9246: { Assigns a value to a variable }
9247: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9248: { Adds a function to the parser }
9249: { ------------------------------------------------------------
9250:   Matrices and linear equations
9251:   ------------------------------------------------------------ }
9252: procedure GaussJordan(A           : TMatrix;
9253:                       Lb, Ub1, Ub2 : Integer;
9254:                       var Det      : Float); external 'dmath';
9255: { Transforms a matrix according to the Gauss-Jordan method }
9256: procedure LinEq(A       : TMatrix;
9257:                 B       : TVector;
9258:                 Lb, Ub  : Integer;
```

```
9259:                 var Det : Float); external 'dmath';
9260: { Solves a linear system according to the Gauss-Jordan method }
9261: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9262: { Cholesky factorization of a positive definite symmetric matrix }
9263: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9264: { LU decomposition }
9265: procedure LU_Solve(A      : TMatrix;
9266:                    B      : TVector;
9267:                    Lb, Ub : Integer;
9268:                    X      : TVector); external 'dmath';
9269: { Solution of linear system from LU decomposition }
9270: procedure QR_Decomp(A            : TMatrix;
9271:                     Lb, Ub1, Ub2 : Integer;
9272:                     R            : TMatrix); external 'dmath';
9273: { QR decomposition }
9274: procedure QR_Solve(Q, R         : TMatrix;
9275:                    B            : TVector;
9276:                    Lb, Ub1, Ub2 : Integer;
9277:                    X            : TVector); external 'dmath';
9278: { Solution of linear system from QR decomposition }
9279: procedure SV_Decomp(A            : TMatrix;
9280:                     Lb, Ub1, Ub2 : Integer;
9281:                     S            : TVector;
9282:                     V            : TMatrix); external 'dmath';
9283: { Singular value decomposition }
9284: procedure SV_SetZero(S      : TVector;
9285:                      Lb, Ub : Integer;
9286:                      Tol    : Float); external 'dmath';
9287: { Set lowest singular values to zero }
9288: procedure SV_Solve(U            : TMatrix;
9289:                    S            : TVector;
9290:                    V            : TMatrix;
9291:                    B            : TVector;
9292:                    Lb, Ub1, Ub2 : Integer;
9293:                    X            : TVector); external 'dmath';
9294: { Solution of linear system from SVD }
9295: procedure SV_Approx(U            : TMatrix;
9296:                     S            : TVector;
9297:                     V            : TMatrix;
9298:                     Lb, Ub1, Ub2 : Integer;
9299:                     A            : TMatrix); external 'dmath';
9300: { Matrix approximation from SVD }
9301: procedure EigenVals(A      : TMatrix;
9302:                     Lb, Ub : Integer;
9303:                     Lambda : TCompVector); external 'dmath';
9304: { Eigenvalues of a general square matrix }
9305: procedure EigenVect(A      : TMatrix;
9306:                     Lb, Ub : Integer;
9307:                     Lambda : TCompVector;
9308:                     V      : TMatrix); external 'dmath';
9309: { Eigenvalues and eigenvectors of a general square matrix }
9310: procedure EigenSym(A      : TMatrix;
9311:                    Lb, Ub : Integer;
9312:                    Lambda : TVector;
9313:                    V      : TMatrix); external 'dmath';
9314: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9315: procedure Jacobi(A              : TMatrix;
9316:                  Lb, Ub, MaxIter : Integer;
9317:                  Tol             : Float;
9318:                  Lambda          : TVector;
9319:                  V               : TMatrix); external 'dmath';
9320: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9321: { ---------------------------------------------------------------
9322:   Optimization
9323:   --------------------------------------------------------------- }
9324: procedure MinBrack(Func                    : TFunc;
9325:                    var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9326: { Brackets a minimum of a function }
9327: procedure GoldSearch(Func           : TFunc;
9328:                      A, B           : Float;
9329:                      MaxIter        : Integer;
9330:                      Tol            : Float;
9331:                      var Xmin, Ymin : Float); external 'dmath';
9332: { Minimization of a function of one variable (golden search) }
9333: procedure LinMin(Func      : TFuncNVar;
9334:                  X, DeltaX : TVector;
9335:                  Lb, Ub    : Integer;
9336:                  var R     : Float;
9337:                  MaxIter   : Integer;
9338:                  Tol       : Float;
9339:                  var F_min : Float); external 'dmath';
9340: { Minimization of a function of several variables along a line }
9341: procedure Newton(Func      : TFuncNVar;
9342:                  HessGrad  : THessGrad;
9343:                  X         : TVector;
9344:                  Lb, Ub    : Integer;
9345:                  MaxIter   : Integer;
9346:                  Tol       : Float;
9347:                  var F_min : Float;
```

```
9348:                 G         : TVector;
9349:                 H_inv     : TMatrix;
9350:                 var Det   : Float); external 'dmath';
9351: { Minimization of a function of several variables (Newton's method) }
9352: procedure SaveNewton(FileName : string); external 'dmath';
9353: { Save Newton iterations in a file }
9354: procedure Marquardt(Func      : TFuncNVar;
9355:                     HessGrad  : THessGrad;
9356:                     X         : TVector;
9357:                     Lb, Ub    : Integer;
9358:                     MaxIter   : Integer;
9359:                     Tol       : Float;
9360:                     var F_min : Float;
9361:                     G         : TVector;
9362:                     H_inv     : TMatrix;
9363:                     var Det   : Float); external 'dmath';
9364: { Minimization of a function of several variables (Marquardt's method) }
9365: procedure SaveMarquardt(FileName : string); external 'dmath';
9366: { Save Marquardt iterations in a file }
9367: procedure BFGS(Func      : TFuncNVar;
9368:                Gradient  : TGradient;
9369:                X         : TVector;
9370:                Lb, Ub    : Integer;
9371:                MaxIter   : Integer;
9372:                Tol       : Float;
9373:                var F_min : Float;
9374:                G         : TVector;
9375:                H_inv     : TMatrix); external 'dmath';
9376: { Minimization of a function of several variables (BFGS method) }
9377: procedure SaveBFGS(FileName : string); external 'dmath';
9378: { Save BFGS iterations in a file }
9379: procedure Simplex(Func      : TFuncNVar;
9380:                   X         : TVector;
9381:                   Lb, Ub    : Integer;
9382:                   MaxIter   : Integer;
9383:                   Tol       : Float;
9384:                   var F_min : Float); external 'dmath';
9385: { Minimization of a function of several variables (Simplex) }
9386: procedure SaveSimplex(FileName : string); external 'dmath';
9387: { Save Simplex iterations in a file }
9388: { ---------------------------------------------------------------
9389:   Nonlinear equations
9390:   --------------------------------------------------------------- }
9391: procedure RootBrack(Func             : TFunc;
9392:                     var X, Y, FX, FY : Float); external 'dmath';
9393: { Brackets a root of function Func between X and Y }
9394: procedure Bisect(Func     : TFunc;
9395:                  var X, Y : Float;
9396:                  MaxIter  : Integer;
9397:                  Tol      : Float;
9398:                  var F    : Float); external 'dmath';
9399: { Bisection method }
9400: procedure Secant(Func     : TFunc;
9401:                  var X, Y : Float;
9402:                  MaxIter  : Integer;
9403:                  Tol      : Float;
9404:                  var F    : Float); external 'dmath';
9405: { Secant method }
9406: procedure NewtEq(Func, Deriv : TFunc;
9407:                  var X       : Float;
9408:                  MaxIter     : Integer;
9409:                  Tol         : Float;
9410:                  var F       : Float); external 'dmath';
9411: { Newton-Raphson method for a single nonlinear equation }
9412: procedure NewtEqs(Equations : TEquations;
9413:                   Jacobian  : TJacobian;
9414:                   X, F      : TVector;
9415:                   Lb, Ub    : Integer;
9416:                   MaxIter   : Integer;
9417:                   Tol       : Float); external 'dmath';
9418: { Newton-Raphson method for a system of nonlinear equations }
9419: procedure Broyden(Equations : TEquations;
9420:                   X, F      : TVector;
9421:                   Lb, Ub    : Integer;
9422:                   MaxIter   : Integer;
9423:                   Tol       : Float); external 'dmath';
9424: { Broyden's method for a system of nonlinear equations }
9425: { ---------------------------------------------------------------
9426:   Polynomials and rational fractions
9427:   --------------------------------------------------------------- }
9428: function Poly(X    : Float;
9429:               Coef : TVector;
9430:               Deg  : Integer) : Float; external 'dmath';
9431: { Evaluates a polynomial }
9432: function RFrac(X          : Float;
9433:                Coef       : TVector;
9434:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9435: { Evaluates a rational fraction }
9436: function RootPol1(A, B  : Float;
```

```
9437:                   var X : Float) : Integer; external 'dmath';
9438: { Solves the linear equation A + B * X = 0 }
9439: function RootPol2(Coef : TVector;
9440:                   Z    : TCompVector) : Integer; external 'dmath';
9441: { Solves a quadratic equation }
9442: function RootPol3(Coef : TVector;
9443:                   Z    : TCompVector) : Integer; external 'dmath';
9444: { Solves a cubic equation }
9445: function RootPol4(Coef : TVector;
9446:                   Z    : TCompVector) : Integer; external 'dmath';
9447: { Solves a quartic equation }
9448: function RootPol(Coef : TVector;
9449:                  Deg  : Integer;
9450:                  Z    : TCompVector) : Integer; external 'dmath';
9451: { Solves a polynomial equation }
9452: function SetRealRoots(Deg : Integer;
9453:                       Z   : TCompVector;
9454:                       Tol : Float) : Integer; external 'dmath';
9455: { Set the imaginary part of a root to zero }
9456: procedure SortRoots(Deg : Integer;
9457:                     Z   : TCompVector); external 'dmath';
9458: { Sorts the roots of a polynomial }
9459: { ------------------------------------------------------------------
9460:   Numerical integration and differential equations
9461:   ------------------------------------------------------------------ }
9462: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9463: { Integration by trapezoidal rule }
9464: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9465: { Integral from A to B }
9466: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9467: { Integral from 0 to B }
9468: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9469: { Convolution product at time T }
9470: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9471: { Convolution by trapezoidal rule }
9472: procedure RKF45(F                    : TDiffEqs;
9473:                 Neqn                 : Integer;
9474:                 Y, Yp                : TVector;
9475:                 var T                : Float;
9476:                 Tout, RelErr, AbsErr : Float;
9477:                 var Flag             : Integer); external 'dmath';
9478: { Integration of a system of differential equations }
9479: { ------------------------------------------------------------------
9480:   Fast Fourier Transform
9481:   ------------------------------------------------------------------ }
9482: procedure FFT(NumSamples        : Integer;
9483:               InArray, OutArray : TCompVector); external 'dmath';
9484: { Fast Fourier Transform }
9485: procedure IFFT(NumSamples        : Integer;
9486:                InArray, OutArray : TCompVector); external 'dmath';
9487: { Inverse Fast Fourier Transform }
9488: procedure FFT_Integer(NumSamples     : Integer;
9489:                       RealIn, ImagIn : TIntVector;
9490:                       OutArray       : TCompVector); external 'dmath';
9491: { Fast Fourier Transform for integer data }
9492: procedure FFT_Integer_Cleanup; external 'dmath';
9493: { Clear memory after a call to FFT_Integer }
9494: procedure CalcFrequency(NumSamples,
9495:                         FrequencyIndex : Integer;
9496:                         InArray        : TCompVector;
9497:                         var FFT        : Complex); external 'dmath';
9498: { Direct computation of Fourier transform }
9499: { ------------------------------------------------------------------
9500:   Random numbers
9501:   ------------------------------------------------------------------ }
9502: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9503: { Select generator }
9504: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9505: { Initialize generator }
9506: function IRanGen : RNG_IntType; external 'dmath';
9507: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9508: function IRanGen31 : RNG_IntType; external 'dmath';
9509: { 31-bit random integer in [0 .. 2^31 - 1] }
9510: function RanGen1 : Float; external 'dmath';
9511: { 32-bit random real in [0,1] }
9512: function RanGen2 : Float; external 'dmath';
9513: { 32-bit random real in [0,1) }
9514: function RanGen3 : Float; external 'dmath';
9515: { 32-bit random real in (0,1) }
9516: function RanGen53 : Float; external 'dmath';
9517: { 53-bit random real in [0,1) }
9518: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9519: { Initializes the 'Multiply with carry' random number generator }
9520: function IRanMWC : RNG_IntType; external 'dmath';
9521: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9522: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9523: { Initializes Mersenne Twister generator with a seed }
9524: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9525:                         KeyLength : Word); external 'dmath';
```

```
9526: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9527: function IRanMT : RNG_IntType; external 'dmath';
9528: { Random integer from MT generator }
9529: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9530: { Initializes the UVAG generator with a string }
9531: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9532: { Initializes the UVAG generator with an integer }
9533: function IRanUVAG : RNG_IntType; external 'dmath';
9534: { Random integer from UVAG generator }
9535: function RanGaussStd : Float; external 'dmath';
9536: { Random number from standard normal distribution }
9537: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9538: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9539: procedure RanMult(M      : TVector; L      : TMatrix;
9540:                   Lb, Ub : Integer;
9541:                   X      : TVector); external 'dmath';
9542: { Random vector from multinormal distribution (correlated) }
9543: procedure RanMultIndep(M, S   : TVector;
9544:                        Lb, Ub : Integer;
9545:                        X      : TVector); external 'dmath';
9546: { Random vector from multinormal distribution (uncorrelated) }
9547: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9548: { Initializes Metropolis-Hastings parameters }
9549: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9550: { Returns Metropolis-Hastings parameters }
9551: procedure Hastings(Func      : TFuncNVar;
9552:                    T         : Float;
9553:                    X         : TVector;
9554:                    V         : TMatrix;
9555:                    Lb, Ub    : Integer;
9556:                    Xmat      : TMatrix;
9557:                    X_min     : TVector;
9558:                    var F_min : Float); external 'dmath';
9559: { Simulation of a probability density function by Metropolis-Hastings }
9560: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9561: { Initializes Simulated Annealing parameters }
9562: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9563: { Initializes log file }
9564: procedure SimAnn(Func          : TFuncNVar;
9565:                  X, Xmin, Xmax : TVector;
9566:                  Lb, Ub        : Integer;
9567:                  var F_min     : Float); external 'dmath';
9568: { Minimization of a function of several var. by simulated annealing }
9569: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9570: { Initializes Genetic Algorithm parameters }
9571: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9572: { Initializes log file }
9573: procedure GenAlg(Func          : TFuncNVar;
9574:                  X, Xmin, Xmax : TVector;
9575:                  Lb, Ub        : Integer;
9576:                  var F_min     : Float); external 'dmath';
9577: { Minimization of a function of several var. by genetic algorithm }
9578: { ---------------------------------------------------------------
9579:   Statistics
9580:   --------------------------------------------------------------- }
9581: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9582: { Mean of sample X }
9583: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9584: { Minimum of sample X }
9585: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9586: { Maximum of sample X }
9587: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9588: { Median of sample X }
9589: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9590: { Standard deviation estimated from sample X }
9591: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9592: { Standard deviation of population }
9593: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9594: { Correlation coefficient }
9595: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9596: { Skewness of sample X }
9597: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9598: { Kurtosis of sample X }
9599: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9600: { Quick sort (ascending order) }
9601: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9602: { Quick sort (descending order) }
9603: procedure Interval(X1, X2             : Float;
9604:                    MinDiv, MaxDiv     : Integer;
9605:                    var Min, Max, Step : Float); external 'dmath';
9606: { Determines an interval for a set of values }
9607: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9608:                     var XMin, XMax, XStep : Float); external 'dmath';
9609: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9610: procedure StudIndep(N1, N2         : Integer;
9611:                     M1, M2, S1, S2 : Float;
9612:                     var T          : Float;
9613:                     var DoF        : Integer); external 'dmath';
9614: { Student t-test for independent samples }
```

```
9615: procedure StudPaired(X, Y    : TVector;
9616:                      Lb, Ub  : Integer;
9617:                      var T   : Float;
9618:                      var DoF : Integer); external 'dmath';
9619: { Student t-test for paired samples }
9620: procedure AnOVa1(Ns               : Integer;
9621:                  N                : TIntVector;
9622:                  M, S             : TVector;
9623:                  var V_f, V_r, F  : Float;
9624:                  var DoF_f, DoF_r : Integer); external 'dmath';
9625: { One-way analysis of variance }
9626: procedure AnOVa2(NA, NB, Nobs : Integer;
9627:                  M, S         : TMatrix;
9628:                  V, F         : TVector;
9629:                  DoF          : TIntVector); external 'dmath';
9630: { Two-way analysis of variance }
9631: procedure Snedecor(N1, N2         : Integer;
9632:                    S1, S2         : Float;
9633:                    var F          : Float;
9634:                    var DoF1, DoF2 : Integer); external 'dmath';
9635: { Snedecor's F-test (comparison of two variances) }
9636: procedure Bartlett(Ns       : Integer;
9637:                    N        : TIntVector;
9638:                    S        : TVector;
9639:                    var Khi2 : Float;
9640:                    var DoF  : Integer); external 'dmath';
9641: { Bartlett's test (comparison of several variances) }
9642: procedure Mann_Whitney(N1, N2     : Integer;
9643:                        X1, X2     : TVector;
9644:                        var U, Eps : Float); external 'dmath';
9645: { Mann-Whitney test}
9646: procedure Wilcoxon(X, Y       : TVector;
9647:                    Lb, Ub     : Integer;
9648:                    var Ndiff  : Integer;
9649:                    var T, Eps : Float); external 'dmath';
9650: { Wilcoxon test }
9651: procedure Kruskal_Wallis(Ns      : Integer;
9652:                          N       : TIntVector;
9653:                          X       : TMatrix;
9654:                          var H   : Float;
9655:                          var DoF : Integer); external 'dmath';
9656: { Kruskal-Wallis test }
9657: procedure Khi2_Conform(N_cls    : Integer;
9658:                        N_estim  : Integer;
9659:                        Obs      : TIntVector;
9660:                        Calc     : TVector;
9661:                        var Khi2 : Float;
9662:                        var DoF  : Integer); external 'dmath';
9663: { Khi-2 test for conformity }
9664: procedure Khi2_Indep(N_lin    : Integer;
9665:                      N_col    : Integer;
9666:                      Obs      : TIntMatrix;
9667:                      var Khi2 : Float;
9668:                      var DoF  : Integer); external 'dmath';
9669: { Khi-2 test for independence }
9670: procedure Woolf_Conform(N_cls   : Integer;
9671:                         N_estim : Integer;
9672:                         Obs     : TIntVector;
9673:                         Calc    : TVector;
9674:                         var G   : Float;
9675:                         var DoF : Integer); external 'dmath';
9676: { Woolf's test for conformity }
9677: procedure Woolf_Indep(N_lin   : Integer;
9678:                       N_col   : Integer;
9679:                       Obs     : TIntMatrix;
9680:                       var G   : Float;
9681:                       var DoF : Integer); external 'dmath';
9682: { Woolf's test for independence }
9683: procedure DimStatClassVector(var C : TStatClassVector;
9684:                              Ub    : Integer); external 'dmath';
9685: { Allocates an array of statistical classes: C[0..Ub] }
9686: procedure Distrib(X       : TVector;
9687:                   Lb, Ub  : Integer;
9688:                   A, B, H : Float;
9689:                   C       : TStatClassVector); external 'dmath';
9690: { Distributes an array X[Lb..Ub] into statistical classes }
9691: { ---------------------------------------------------------------
9692:   Linear / polynomial regression
9693:   ---------------------------------------------------------------- }
9694: procedure LinFit(X, Y   : TVector;
9695:                  Lb, Ub : Integer;
9696:                  B      : TVector;
9697:                  V      : TMatrix); external 'dmath';
9698: { Linear regression : Y = B(0) + B(1) * X }
9699: procedure WLinFit(X, Y, S : TVector;
9700:                   Lb, Ub  : Integer;
9701:                   B       : TVector;
9702:                   V       : TMatrix); external 'dmath';
9703: { Weighted linear regression : Y = B(0) + B(1) * X }
```

```
9704: procedure SVDLinFit(X, Y   : TVector;
9705:                     Lb, Ub : Integer;
9706:                     SVDTol : Float;
9707:                     B      : TVector;
9708:                     V      : TMatrix); external 'dmath';
9709: { Unweighted linear regression by singular value decomposition }
9710: procedure WSVDLinFit(X, Y, S : TVector;
9711:                      Lb, Ub  : Integer;
9712:                      SVDTol  : Float;
9713:                      B       : TVector;
9714:                      V       : TMatrix); external 'dmath';
9715: { Weighted linear regression by singular value decomposition }
9716: procedure MulFit(X            : TMatrix;
9717:                  Y            : TVector;
9718:                  Lb, Ub, Nvar : Integer;
9719:                  ConsTerm     : Boolean;
9720:                  B            : TVector;
9721:                  V            : TMatrix); external 'dmath';
9722: { Multiple linear regression by Gauss-Jordan method }
9723: procedure WMulFit(X            : TMatrix;
9724:                   Y, S         : TVector;
9725:                   Lb, Ub, Nvar : Integer;
9726:                   ConsTerm     : Boolean;
9727:                   B            : TVector;
9728:                   V            : TMatrix); external 'dmath';
9729: { Weighted multiple linear regression by Gauss-Jordan method }
9730: procedure SVDFit(X            : TMatrix;
9731:                  Y            : TVector;
9732:                  Lb, Ub, Nvar : Integer;
9733:                  ConsTerm     : Boolean;
9734:                  SVDTol       : Float;
9735:                  B            : TVector;
9736:                  V            : TMatrix); external 'dmath';
9737: { Multiple linear regression by singular value decomposition }
9738: procedure WSVDFit(X            : TMatrix;
9739:                   Y, S         : TVector;
9740:                   Lb, Ub, Nvar : Integer;
9741:                   ConsTerm     : Boolean;
9742:                   SVDTol       : Float;
9743:                   B            : TVector;
9744:                   V            : TMatrix); external 'dmath';
9745: { Weighted multiple linear regression by singular value decomposition }
9746: procedure PolFit(X, Y        : TVector;
9747:                  Lb, Ub, Deg : Integer;
9748:                  B           : TVector;
9749:                  V           : TMatrix); external 'dmath';
9750: { Polynomial regression by Gauss-Jordan method }
9751: procedure WPolFit(X, Y, S     : TVector;
9752:                   Lb, Ub, Deg : Integer;
9753:                   B           : TVector;
9754:                   V           : TMatrix); external 'dmath';
9755: { Weighted polynomial regression by Gauss-Jordan method }
9756: procedure SVDPolFit(X, Y        : TVector;
9757:                     Lb, Ub, Deg : Integer;
9758:                     SVDTol      : Float;
9759:                     B           : TVector;
9760:                     V           : TMatrix); external 'dmath';
9761: { Unweighted polynomial regression by singular value decomposition }
9762: procedure WSVDPolFit(X, Y, S     : TVector;
9763:                      Lb, Ub, Deg : Integer;
9764:                      SVDTol      : Float;
9765:                      B           : TVector;
9766:                      V           : TMatrix); external 'dmath';
9767: { Weighted polynomial regression by singular value decomposition }
9768: procedure RegTest(Y, Ycalc : TVector;
9769:                   LbY, UbY : Integer;
9770:                   V        : TMatrix;
9771:                   LbV, UbV : Integer;
9772:                   var Test : TRegTest); external 'dmath';
9773: { Test of unweighted regression }
9774: procedure WRegTest(Y, Ycalc, S : TVector;
9775:                    LbY, UbY    : Integer;
9776:                    V           : TMatrix;
9777:                    LbV, UbV    : Integer;
9778:                    var Test    : TRegTest); external 'dmath';
9779: { Test of weighted regression }
9780: { ---------------------------------------------------------------
9781:   Nonlinear regression
9782:   --------------------------------------------------------------- }
9783: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9784: { Sets the optimization algorithm for nonlinear regression }
9785: function GetOptAlgo : TOptAlgo; external 'dmath';
9786: { Returns the optimization algorithm }
9787: procedure SetMaxParam(N : Byte); external 'dmath';
9788: { Sets the maximum number of regression parameters for nonlinear regression }
9789: function GetMaxParam : Byte; external 'dmath';
9790: { Returns the maximum number of regression parameters for nonlinear regression }
9791: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9792: { Sets the bounds on the I-th regression parameter }
```

```
9793: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9794: { Returns the bounds on the I-th regression parameter }
9795: procedure NLFit(RegFunc   : TRegFunc;
9796:                 DerivProc : TDerivProc;
9797:                 X, Y      : TVector;
9798:                 Lb, Ub    : Integer;
9799:                 MaxIter   : Integer;
9800:                 Tol       : Float;
9801:                 B         : TVector;
9802:                 FirstPar,
9803:                 LastPar   : Integer;
9804:                 V         : TMatrix); external 'dmath';
9805: { Unweighted nonlinear regression }
9806: procedure WNLFit(RegFunc   : TRegFunc;
9807:                  DerivProc : TDerivProc;
9808:                  X, Y, S   : TVector;
9809:                  Lb, Ub    : Integer;
9810:                  MaxIter   : Integer;
9811:                  Tol       : Float;
9812:                  B         : TVector;
9813:                  FirstPar,
9814:                  LastPar   : Integer;
9815:                  V         : TMatrix); external 'dmath';
9816: { Weighted nonlinear regression }
9817: procedure SetMCFile(FileName : String); external 'dmath';
9818: { Set file for saving MCMC simulations }
9819: procedure SimFit(RegFunc   : TRegFunc;
9820:                  X, Y      : TVector;
9821:                  Lb, Ub    : Integer;
9822:                  B         : TVector;
9823:                  FirstPar,
9824:                  LastPar   : Integer;
9825:                  V         : TMatrix); external 'dmath';
9826: { Simulation of unweighted nonlinear regression by MCMC }
9827: procedure WSimFit(RegFunc   : TRegFunc;
9828:                   X, Y, S   : TVector;
9829:                   Lb, Ub    : Integer;
9830:                   B         : TVector;
9831:                   FirstPar,
9832:                   LastPar   : Integer;
9833:                   V         : TMatrix); external 'dmath';
9834: { Simulation of weighted nonlinear regression by MCMC }
9835: { ------------------------------------------------------------------
9836:   Nonlinear regression models
9837:   ------------------------------------------------------------------ }
9838: procedure FracFit(X, Y       : TVector;
9839:                   Lb, Ub     : Integer;
9840:                   Deg1, Deg2 : Integer;
9841:                   ConsTerm   : Boolean;
9842:                   MaxIter    : Integer;
9843:                   Tol        : Float;
9844:                   B          : TVector;
9845:                   V          : TMatrix); external 'dmath';
9846: { Unweighted fit of rational fraction }
9847: procedure WFracFit(X, Y, S    : TVector;
9848:                    Lb, Ub     : Integer;
9849:                    Deg1, Deg2 : Integer;
9850:                    ConsTerm   : Boolean;
9851:                    MaxIter    : Integer;
9852:                    Tol        : Float;
9853:                    B          : TVector;
9854:                    V          : TMatrix); external 'dmath';
9855: { Weighted fit of rational fraction }
9856:
9857: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9858: { Returns the value of the rational fraction at point X }
9859: procedure ExpFit(X, Y         : TVector;
9860:                  Lb, Ub, Nexp : Integer;
9861:                  ConsTerm     : Boolean;
9862:                  MaxIter      : Integer;
9863:                  Tol          : Float;
9864:                  B            : TVector;
9865:                  V            : TMatrix); external 'dmath';
9866: { Unweighted fit of sum of exponentials }
9867: procedure WExpFit(X, Y, S      : TVector;
9868:                   Lb, Ub, Nexp : Integer;
9869:                   ConsTerm     : Boolean;
9870:                   MaxIter      : Integer;
9871:                   Tol          : Float;
9872:                   B            : TVector;
9873:                   V            : TMatrix); external 'dmath';
9874: { Weighted fit of sum of exponentials }
9875: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9876: { Returns the value of the regression function at point X }
9877: procedure IncExpFit(X, Y     : TVector;
9878:                     Lb, Ub   : Integer;
9879:                     ConsTerm : Boolean;
9880:                     MaxIter  : Integer;
9881:                     Tol      : Float;
```

```
9882:                     B        : TVector;
9883:                     V        : TMatrix); external 'dmath';
9884: { Unweighted fit of model of increasing exponential }
9885: procedure WIncExpFit(X, Y, S  : TVector;
9886:                      Lb, Ub   : Integer;
9887:                      ConsTerm : Boolean;
9888:                      MaxIter  : Integer;
9889:                      Tol      : Float;
9890:                      B        : TVector;
9891:                      V        : TMatrix); external 'dmath';
9892: { Weighted fit of increasing exponential }
9893: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9894: { Returns the value of the regression function at point X }
9895: procedure ExpLinFit(X, Y    : TVector;
9896:                     Lb, Ub  : Integer;
9897:                     MaxIter : Integer;
9898:                     Tol     : Float;
9899:                     B       : TVector;
9900:                     V       : TMatrix); external 'dmath';
9901: { Unweighted fit of the "exponential + linear" model }
9902: procedure WExpLinFit(X, Y, S : TVector;
9903:                      Lb, Ub  : Integer;
9904:                      MaxIter : Integer;
9905:                      Tol     : Float;
9906:                      B       : TVector;
9907:                      V       : TMatrix); external 'dmath';
9908: { Weighted fit of the "exponential + linear" model }
9909:
9910: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9911: { Returns the value of the regression function at point X }
9912: procedure MichFit(X, Y    : TVector;
9913:                   Lb, Ub  : Integer;
9914:                   MaxIter : Integer;
9915:                   Tol     : Float;
9916:                   B       : TVector;
9917:                   V       : TMatrix); external 'dmath';
9918: { Unweighted fit of Michaelis equation }
9919: procedure WMichFit(X, Y, S : TVector;
9920:                    Lb, Ub  : Integer;
9921:                    MaxIter : Integer;
9922:                    Tol     : Float;
9923:                    B       : TVector;
9924:                    V       : TMatrix); external 'dmath';
9925: { Weighted fit of Michaelis equation }
9926: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9927: { Returns the value of the Michaelis equation at point X }
9928: procedure MintFit(X, Y    : TVector;
9929:                   Lb, Ub  : Integer;
9930:                   MintVar : TMintVar;
9931:                   Fit_S0  : Boolean;
9932:                   MaxIter : Integer;
9933:                   Tol     : Float;
9934:                   B       : TVector;
9935:                   V       : TMatrix); external 'dmath';
9936: { Unweighted fit of the integrated Michaelis equation }
9937: procedure WMintFit(X, Y, S : TVector;
9938:                    Lb, Ub  : Integer;
9939:                    MintVar : TMintVar;
9940:                    Fit_S0  : Boolean;
9941:                    MaxIter : Integer;
9942:                    Tol     : Float;
9943:                    B       : TVector;
9944:                    V       : TMatrix); external 'dmath';
9945: { Weighted fit of the integrated Michaelis equation }
9946: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9947: { Returns the value of the integrated Michaelis equation at point X }
9948: procedure HillFit(X, Y    : TVector;
9949:                   Lb, Ub  : Integer;
9950:                   MaxIter : Integer;
9951:                   Tol     : Float;
9952:                   B       : TVector;
9953:                   V       : TMatrix); external 'dmath';
9954: { Unweighted fit of Hill equation }
9955: procedure WHillFit(X, Y, S : TVector;
9956:                    Lb, Ub  : Integer;
9957:                    MaxIter : Integer;
9958:                    Tol     : Float;
9959:                    B       : TVector;
9960:                    V       : TMatrix); external 'dmath';
9961: { Weighted fit of Hill equation }
9962: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9963: { Returns the value of the Hill equation at point X }
9964: procedure LogiFit(X, Y     : TVector;
9965:                   Lb, Ub   : Integer;
9966:                   ConsTerm : Boolean;
9967:                   General  : Boolean;
9968:                   MaxIter  : Integer;
9969:                   Tol      : Float;
9970:                   B        : TVector;
```

```
 9971:                   V       : TMatrix); external 'dmath';
 9972: { Unweighted fit of logistic function }
 9973: procedure WLogiFit(X, Y, S  : TVector;
 9974:                    Lb, Ub   : Integer;
 9975:                    ConsTerm : Boolean;
 9976:                    General  : Boolean;
 9977:                    MaxIter  : Integer;
 9978:                    Tol      : Float;
 9979:                    B        : TVector;
 9980:                    V        : TMatrix); external 'dmath';
 9981: { Weighted fit of logistic function }
 9982: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9983: { Returns the value of the logistic function at point X }
 9984: procedure PKFit(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 the acid-base titration curve }
 9991: procedure WPKFit(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 the acid-base titration curve }
 9998: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9999: { Returns the value of the acid-base titration function at point X }
10000: procedure PowFit(X, Y    : TVector;
10001:                  Lb, Ub  : Integer;
10002:                  MaxIter : Integer;
10003:                  Tol     : Float;
10004:                  B       : TVector;
10005:                  V       : TMatrix); external 'dmath';
10006: { Unweighted fit of power function }
10007: procedure WPowFit(X, Y, S : TVector;
10008:                   Lb, Ub  : Integer;
10009:                   MaxIter : Integer;
10010:                   Tol     : Float;
10011:                   B       : TVector;
10012:                   V       : TMatrix); external 'dmath';
10013: { Weighted fit of power function }
10014:
10015: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10016: { Returns the value of the power function at point X }
10017: procedure GammaFit(X, Y    : TVector;
10018:                    Lb, Ub  : Integer;
10019:                    MaxIter : Integer;
10020:                    Tol     : Float;
10021:                    B       : TVector;
10022:                    V       : TMatrix); external 'dmath';
10023: { Unweighted fit of gamma distribution function }
10024: procedure WGammaFit(X, Y, S : TVector;
10025:                     Lb, Ub  : Integer;
10026:                     MaxIter : Integer;
10027:                     Tol     : Float;
10028:                     B       : TVector;
10029:                     V       : TMatrix); external 'dmath';
10030: { Weighted fit of gamma distribution function }
10031: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10032: { Returns the value of the gamma distribution function at point X }
10033: { ---------------------------------------------------------------
10034:   Principal component analysis
10035:   --------------------------------------------------------------- }
10036: procedure VecMean(X           : TMatrix;
10037:                   Lb, Ub, Nvar : Integer;
10038:                   M           : TVector); external 'dmath';
10039: { Computes the mean vector M from matrix X }
10040: procedure VecSD(X            : TMatrix;
10041:                 Lb, Ub, Nvar : Integer;
10042:                 M, S         : TVector); external 'dmath';
10043: { Computes the vector of standard deviations S from matrix X }
10044: procedure MatVarCov(X            : TMatrix;
10045:                     Lb, Ub, Nvar : Integer;
10046:                     M            : TVector;
10047:                     V            : TMatrix); external 'dmath';
10048: { Computes the variance-covariance matrix V from matrix X }
10049: procedure MatCorrel(V    : TMatrix;
10050:                     Nvar : Integer;
10051:                     R    : TMatrix); external 'dmath';
10052: { Computes the correlation matrix R from the var-cov matrix V }
10053: procedure PCA(R      : TMatrix;
10054:               Nvar   : Integer;
10055:               Lambda : TVector;
10056:               C, Rc  : TMatrix); external 'dmath';
10057: { Performs a principal component analysis of the correlation matrix R }
10058: procedure ScaleVar(X           : TMatrix;
10059:                    Lb, Ub, Nvar : Integer;
```

```
10060:                   M, S         : TVector;
10061:                   Z            : TMatrix); external 'dmath';
10062: { Scales a set of variables by subtracting means and dividing by SD's }
10063: procedure PrinFac(Z            : TMatrix;
10064:                   Lb, Ub, Nvar : Integer;
10065:                   C, F         : TMatrix); external 'dmath';
10066: { Computes principal factors }
10067: { ---------------------------------------------------------------
10068:   Strings
10069:   --------------------------------------------------------------- }
10070: function LTrim(S : String) : String; external 'dmath';
10071: { Removes leading blanks }
10072: function RTrim(S : String) : String; external 'dmath';
10073: { Removes trailing blanks }
10074: function Trim(S : String) : String; external 'dmath';
10075: { Removes leading and trailing blanks }
10076: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10077: { Returns a string made of character C repeated N times }
10078: function RFill(S : String; L : Byte) : String; external 'dmath';
10079: { Completes string S with trailing blanks for a total length L }
10080: function LFill(S : String; L : Byte) : String; external 'dmath';
10081: { Completes string S with leading blanks for a total length L }
10082: function CFill(S : String; L : Byte) : String; external 'dmath';
10083: { Centers string S on a total length L }
10084: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10085: { Replaces in string S all the occurences of C1 by C2 }
10086: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10087: { Extracts a field from a string }
10088: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10089: { Parses a string into its constitutive fields }
10090: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10091: { Sets the numeric format }
10092: function FloatStr(X : Float) : String; external 'dmath';
10093: { Converts a real to a string according to the numeric format }
10094: function IntStr(N : LongInt) : String; external 'dmath';
10095: { Converts an integer to a string }
10096: function CompStr(Z : Complex) : String; external 'dmath';
10097: { Converts a complex number to a string }
10098: {$IFDEF DELPHI}
10099: function StrDec(S : String) : String; external 'dmath';
10100: { Set decimal separator to the symbol defined in SysUtils }
10101: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10102: { Test if a string represents a number and returns it in X }
10103: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10104: { Reads a floating point number from an Edit control }
10105: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10106: { Writes a floating point number in a text file }
10107: {$ENDIF}
10108: { ---------------------------------------------------------------
10109:   BGI / Delphi graphics
10110:   --------------------------------------------------------------- }
10111: function InitGraphics
10112: {$IFDEF DELPHI}
10113: (Width, Height : Integer) : Boolean;
10114: {$ELSE}
10115: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10116: { Enters graphic mode }
10117: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10118:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10119: { Sets the graphic window }
10120: procedure SetOxScale(Scale               : TScale;
10121:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10122: { Sets the scale on the Ox axis }
10123: procedure SetOyScale(Scale               : TScale;
10124:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10125: { Sets the scale on the Oy axis }
10126: procedure GetOxScale(var Scale               : TScale;
10127:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10128: { Returns the scale on the Ox axis }
10129: procedure GetOyScale(var Scale               : TScale;
10130:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10131: { Returns the scale on the Oy axis }
10132: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10133: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10134: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10135: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10136: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10137: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10138: {$IFNDEF DELPHI}
10139: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10140: { Sets the font for the main graph title }
10141: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10142: { Sets the font for the Ox axis (title and labels) }
10143: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10144: { Sets the font for the Oy axis (title and labels) }
10145: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10146: { Sets the font for the legends }
10147: procedure SetClipping(Clip : Boolean); external 'dmath';
10148: { Determines whether drawings are clipped at the current viewport
```

```
10149:   boundaries, according to the value of the Boolean parameter Clip }
10150: {$ENDIF}
10151: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10152: { Plots the horizontal axis }
10153: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10154: { Plots the vertical axis }
10155: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10156: { Plots a grid on the graph }
10157: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10158: { Writes the title of the graph }
10159: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10160: { Sets the maximum number of curves and re-initializes their parameters }
10161: procedure SetPointParam
10162: {$IFDEF DELPHI}
10163: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10164: {$ELSE}
10165: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10166: { Sets the point parameters for curve # CurvIndex }
10167: procedure SetLineParam
10168: {$IFDEF DELPHI}
10169: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10170: {$ELSE}
10171: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10172: { Sets the line parameters for curve # CurvIndex }
10173: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10174: { Sets the legend for curve # CurvIndex }
10175: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10176: { Sets the step for curve # CurvIndex }
10177: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10178: procedure GetPointParam
10179: {$IFDEF DELPHI}
10180: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10181: {$ELSE}
10182: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10183: { Returns the point parameters for curve # CurvIndex }
10184: procedure GetLineParam
10185: {$IFDEF DELPHI}
10186: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10187: {$ELSE}
10188: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10189: { Returns the line parameters for curve # CurvIndex }
10190: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10191: { Returns the legend for curve # CurvIndex }
10192: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10193: { Returns the step for curve # CurvIndex }
10194: {$IFDEF DELPHI}
10195: procedure PlotPoint(Canvas    : TCanvas;
10196:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
10197: {$ELSE}
10198: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10199: {$ENDIF}
10200: { Plots a point on the screen }
10201: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10202:                     X, Y                  : TVector;
10203:                     Lb, Ub, CurvIndex     : Integer); external 'dmath';
10204: { Plots a curve }
10205: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10206:                                  X, Y, S           : TVector;
10207:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10208: { Plots a curve with error bars }
10209: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10210:                    Func                : TFunc;
10211:                    Xmin, Xmax          : Float;
10212:                    {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10213:                    CurvIndex           : Integer); external 'dmath';
10214: { Plots a function }
10215: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10216:                       NCurv                : Integer;
10217:                       ShowPoints, ShowLines : Boolean); external 'dmath';
10218: { Writes the legends for the plotted curves }
10219: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10220:                  Nx, Ny, Nc           : Integer;
10221:                  X, Y, Z              : TVector;
10222:                  F                    : TMatrix); external 'dmath';
10223: { Contour plot }
10224: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10225: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10226: function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
10227: function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
10228: {$IFNDEF DELPHI}
10229: procedure LeaveGraphics; external 'dmath';
10230: { Quits graphic mode }
10231: {$ENDIF}
10232: { ---------------------------------------------------------------
10233:   LaTeX graphics
10234:   --------------------------------------------------------------- }
10235: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
10236:                           Header            : Boolean) : Boolean; external 'dmath';
10237: { Initializes the LaTeX file }
```

```
10238: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10239: { Sets the graphic window }
10240: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10241: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10242: { Sets the scale on the Ox axis }
10243: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10244: { Sets the scale on the Oy axis }
10245: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10246: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10247: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10248: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10249: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10250: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10251: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10252: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10253: { Sets the maximum number of curves and re-initializes their parameters }
10254: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10255: { Sets the point parameters for curve # CurvIndex }
10256: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10257:                            Width : Float; Smooth : Boolean); external 'dmath';
10258: { Sets the line parameters for curve # CurvIndex }
10259: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10260: { Sets the legend for curve # CurvIndex }
10261: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10262: { Sets the step for curve # CurvIndex }
10263: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10264: { Plots a curve }
10265: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10266:                                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10267: { Plots a curve with error bars }
10268: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10269:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
10270: { Plots a function }
10271: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10272: { Writes the legends for the plotted curves }
10273: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
10274: { Contour plot }
10275: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10276: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10277:
10278: //******************************************************unit uPSI_SynPdf;
10279:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10280:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10281:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10282:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10283:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10284:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10285:  //Function _GetCharCount( Text : PAnsiChar) : integer
10286:  //Procedure L2R( W : PWideChar; L : integer)
10287:  Function PdfCoord( MM : single) : integer
10288:  Function CurrentPrinterPaperSize : TPDFPaperSize
10289:  Function CurrentPrinterRes : TPoint
10290:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10291:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10292:  Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10293:  Const('Usp10','String 'usp10.dll
10294:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10295:    'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10296:   AddTypeS('TScriptState_set', 'set of TScriptState_enum
10297: //*********************************************************************
10298:
10299: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
10300: begin
10301:  Procedure PMrandomize( I : word)
10302:  Function PMrandom : longint
10303:  Function Rrand : extended
10304:  Function Irand( N : word) : word
10305:  Function Brand( P : extended) : boolean
10306:  Function Nrand : extended
10307: end;
10308:
10309: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10310: begin
10311:   Function Endian( x : LongWord) : LongWord
10312:   Function Endian64( x : Int64) : Int64
10313:   Function spRol( x : LongWord; y : Byte) : LongWord
10314:   Function spRor( x : LongWord; y : Byte) : LongWord
10315:   Function Ror64( x : Int64; y : Byte) : Int64
10316: end;
10317:
10318: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10319: begin
10320:  Procedure ClearModules
10321:  Procedure ReadMapFile( Fname : string)
10322:  Function AddressInfo( Address : dword) : string
10323: end;
10324:
10325: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10326: begin
```

```
10327:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10328:    +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10329:    +'teByOther, tpExecuteByOther )
10330:   TTarPermissions', 'set of TTarPermission
10331:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10332:    +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10333:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10334:   TTarModes', 'set of TTarMode
10335:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10336:    +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10337:    +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10338:    +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10339:    +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10340:   SIRegister_TTarArchive(CL);
10341:   SIRegister_TTarWriter(CL);
10342:  Function PermissionString( Permissions : TTarPermissions) : STRING
10343:  Function ConvertFilename( Filename : STRING) : STRING
10344:  Function FileTimeGMT( FileName : STRING) : TDateTime;
10345:  Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10346:  Procedure ClearDirRec( var DirRec : TTarDirRec)
10347: end;
10348:
10349:
10350: //************************************************unit uPSI_TlHelp32;
10351: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10352: begin
10353:  Const('MAX_MODULE_NAME32','LongInt'( 255);
10354:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10355:  Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001);
10356:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10357:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10358:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10359:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10360:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10361:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10362:   AddTypeS('THeapList32', 'tagHEAPLIST32
10363:  Const('HF32_DEFAULT','LongInt'( 1);
10364:  Const('HF32_SHARED','LongInt'( 2);
10365:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10366:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10367:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10368:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10369:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10370:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10371:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10372:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10373:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10374:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10375:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10376:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10377:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10378:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10379:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10380:    +'aPri : Longint; dwFlags : DWORD; end
10381:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10382:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10383:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10384:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10385: end;
10386:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10387:  Const('EW_REBOOTSYSTEM','LongWord( $0043);
10388:  Const('EW_EXITANDEXECAPP','LongWord( $0044);
10389:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10390:  Const('EWX_LOGOFF','LongInt'( 0);
10391:  Const('EWX_SHUTDOWN','LongInt'( 1);
10392:  Const('EWX_REBOOT','LongInt'( 2);
10393:  Const('EWX_FORCE','LongInt'( 4);
10394:  Const('EWX_POWEROFF','LongInt'( 8);
10395:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10396:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10397:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10398:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10399:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10400:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10401:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10402:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10403:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10404:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10405:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10406:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10407:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10408:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10409:  Function GetDesktopWindow : HWND
10410:  Function GetParent( hWnd : HWND) : HWND
10411:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10412:  Function GetTopWindow( hWnd : HWND) : HWND
10413:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10414:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10415:  //Delphi DFM
```

```
10416:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10417:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10418:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10419:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10420:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10421:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10422:  Function OpenIcon( hWnd : HWND) : BOOL
10423:  Function CloseWindow( hWnd : HWND) : BOOL
10424:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10425:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10426:  Function IsWindowVisible( hWnd : HWND) : BOOL
10427:  Function IsIconic( hWnd : HWND) : BOOL
10428:  Function AnyPopup : BOOL
10429:  Function BringWindowToTop( hWnd : HWND) : BOOL
10430:  Function IsZoomed( hWnd : HWND) : BOOL
10431:  Function IsWindow( hWnd : HWND) : BOOL
10432:  Function IsMenu( hMenu : HMENU) : BOOL
10433:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10434:  Function DestroyWindow( hWnd : HWND) : BOOL
10435:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10436:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10437:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
10438:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10439:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10440:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10441:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10442:
10443: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10444: begin
10445:  const('ShowSetupDialogOptLong','String '--setup
10446:  PrimaryConfPathOptLong','String '--primary-config-path=
10447:  PrimaryConfPathOptShort','String '--pcp=
10448:  SecondaryConfPathOptLong','String '--secondary-config-path=
10449:  SecondaryConfPathOptShort','String '--scp=
10450:  NoSplashScreenOptLong','String '--no-splash-screen
10451:  NoSplashScreenOptShort','String '--nsc
10452:  StartedByStartLazarusOpt','String '--started-by-startlazarus
10453:  SkipLastProjectOpt','String '--skip-last-project
10454:  DebugLogOpt','String '--debug-log=
10455:  DebugLogOptEnable','String '--debug-enable=
10456:  LanguageOpt','String '--language=
10457:  LazarusDirOpt','String '--lazarusdir=
10458:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10459:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10460:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10461:  Function IsHelpRequested : Boolean
10462:  Function IsVersionRequested : boolean
10463:  Function GetLanguageSpecified : string
10464:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10465:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10466:  Procedure ParseNoGuiCmdLineParams
10467:  Function ExtractCmdLineFilenames : TStrings
10468: end;
10469:
10470:
10471: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10472: begin
10473:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10474:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10475:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10476:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10477:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10478:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10479:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10480:  Function DirPathExists( DirectoryName : string) : boolean
10481:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10482:  Function ExtractFileNameOnly( const AFilename : string) : string
10483:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10484:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10485:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10486:  Function ForceDirectory( DirectoryName : string) : boolean
10487:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10488:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10489:  Function FileIsText( const AFilename : string) : boolean
10490:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10491:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10492:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10493:  Function TrimFilename( const AFilename : string) : string
10494:  Function ResolveDots( const AFilename : string) : string
10495:  Procedure ForcePathDelims( var FileName : string)
10496:  Function GetForcedPathDelims( const FileName : string) : String
10497:  Function CleanAndExpandFilename( const Filename : string) : string
10498:  Function CleanAndExpandDirectory( const Filename : string) : string
10499:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10500:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10501:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10502:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
```

```
10503:  Function FileIsInPath( const Filename, Path : string) : boolean
10504:  Function AppendPathDelim( const Path : string) : string
10505:  Function ChompPathDelim( const Path : string) : string
10506:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10507:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10508:  Function MinimizeSearchPath( const SearchPath : string) : string
10509:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10510:  (*Function FileExistsUTF8( const Filename : string) : boolean
10511:  Function FileAgeUTF8( const FileName : string) : Longint
10512:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10513:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10514:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10515:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10516:  Procedure FindCloseUTF8( var F : TSearchrec)
10517:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10518:  Function FileGetAttrUTF8( const FileName : String) : Longint
10519:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10520:  Function DeleteFileUTF8( const FileName : String) : Boolean
10521:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10522:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10523:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10524:  Function GetCurrentDirUTF8 : String
10525:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10526:  Function CreateDirUTF8( const NewDir : String) : Boolean
10527:  Function RemoveDirUTF8( const Dir : String) : Boolean
10528:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10529:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10530:  Function FileCreateUTF8( const FileName : string) : THandle;
10531:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10532:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10533:  Function FileSizeUtf8( const Filename : string) : int64
10534:  Function GetFileDescription( const AFilename : string) : string
10535:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10536:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10537:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10538:  Function IsUNCPath( const Path : String) : Boolean
10539:  Function ExtractUNCVolume( const Path : String) : String
10540:  Function ExtractFileRoot( FileName : String) : String
10541:  Function GetDarwinSystemFilename( Filename : string) : string
10542:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10543:  Function StrToCmdLineParam( const Param : string) : string
10544:  Function MergeCmdLineParams( ParamList : TStrings) : string
10545:  Procedure InvalidateFileStateCache( const Filename : string)
10546:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10547:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10548:  Function ReadFileToString( const Filename : string) : string
10549:  type
10550:   TCopyFileFlag = ( cffOverwriteFile,
10551:     cffCreateDestDirectory, cffPreserveTime );
10552:   TCopyFileFlags = set of TCopyFileFlag;*)
10553:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10554:   TCopyFileFlags', 'set of TCopyFileFlag
10555:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10556: end;
10557:
10558: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10559: begin
10560:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10561:   SIRegister_TMask(CL);
10562:   SIRegister_TParseStringList(CL);
10563:   SIRegister_TMaskList(CL);
10564:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10565:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10566:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10567:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10568: end;
10569:
10570: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10571: begin
10572:   //PShellHookInfo', '^TShellHookInfo // will not work
10573:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10574:   SHELLHOOKINFO', 'TShellHookInfo
10575:   LPSHELLHOOKINFO', 'PShellHookInfo
10576:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10577:   SIRegister_TJvShellHook(CL);
10578:  Function InitJvShellHooks : Boolean
10579:  Procedure UnInitJvShellHooks
10580: end;
10581:
10582: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10583: begin
10584:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10585:    +', dcHasSetSel, dcWantTab, dcNative )
10586:   TDlgCodes', 'set of TDlgCode
10587:  'dcWantMessage',' dcWantAllKeys);
10588:   SIRegister_IJvExControl(CL);
10589:   SIRegister_IJvDenySubClassing(CL);
10590:   SIRegister_TStructPtrMessage(CL);
10591:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
```

```
10592:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10593:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10594:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10595:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10596:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10597:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10598:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10599:  Function GetFocusedControl( AControl : TControl) : TWinControl
10600:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10601:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10602:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10603:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10604:   SIRegister_TJvExControl(CL);
10605:   SIRegister_TJvExWinControl(CL);
10606:   SIRegister_TJvExCustomControl(CL);
10607:   SIRegister_TJvExGraphicControl(CL);
10608:   SIRegister_TJvExHintWindow(CL);
10609:   SIRegister_TJvExPubGraphicControl(CL);
10610: end;
10611:
10612: (*---------------------------------------------------------------------------*)
10613: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10614: begin
10615:  Procedure EncodeStream( Input, Output : TStream)
10616:  Procedure DecodeStream( Input, Output : TStream)
10617:  Function EncodeString1( const Input : string) : string
10618:  Function DecodeString1( const Input : string) : string
10619: end;
10620:
10621: (*---------------------------------------------------------------------------*)
10622: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10623: begin
10624:   SIRegister_TWebAppRegInfo(CL);
10625:   SIRegister_TWebAppRegList(CL);
10626:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10627:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10628:  Procedure UnregisterWebApp( const AProgID : string)
10629:  Function FindRegisteredWebApp( const AProgID : string) : string
10630:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10631:  'sUDPPort','String 'UDPPort
10632: end;
10633:
10634: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10635: begin
10636:  // TStringDynArray', 'array of string
10637:  Function GetEnvVarValue( const VarName : string) : string
10638:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10639:  Function DeleteEnvVar( const VarName : string) : Integer
10640:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10641:  Function ExpandEnvVars( const Str : string) : string
10642:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10643:  Procedure GetAllEnvVarNames( const Names : TStrings);
10644:  Function GetAllEnvVarNames1 : TStringDynArray;
10645:  Function EnvBlockSize : Integer
10646:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10647:   SIRegister_TPJEnvVarsEnumerator(CL);
10648:   SIRegister_TPJEnvVars(CL);
10649:   FindClass('TOBJECT'),'EPJEnvVars
10650:   FindClass('TOBJECT'),'EPJEnvVars
10651:  //Procedure Register
10652: end;
10653:
10654: (*---------------------------------------------------------------------------*)
10655: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10656: begin
10657:  'cOneSecInMS','LongInt'( 1000);
10658:  //'cDefTimeSlice','LongInt'( 50);
10659:  //'cDefMaxExecTime',' cOneMinInMS);
10660:  'cAppErrorMask','LongInt'( 1 shl 29);
10661:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10662:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10663:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10664:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10665:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10666:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10667:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10668:   SIRegister_TPJCustomConsoleApp(CL);
10669:   SIRegister_TPJConsoleApp(CL);
10670: end;
10671:
10672: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10673: begin
10674:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10675:   t_encoding', '( uuencode, base64, mime )
10676:  Function internet_date( date : TDateTime) : string
10677:  Function lookup_hostname( const hostname : string) : longint
10678:  Function my_hostname : string
10679:  Function my_ip_address : longint
```

```
10680:  Function ip2string( ip_address : longint) : string
10681:  Function resolve_hostname( ip : longint) : string
10682:  Function address_from( const s : string; count : integer) : string
10683:  Function encode_base64( data : TStream) : TStringList
10684:  Function decode_base64( source : TStringList) : TMemoryStream
10685:  Function posn( const s, t : string; count : integer) : integer
10686:  Function poscn( c : char; const s : string; n : integer) : integer
10687:  Function filename_of( const s : string) : string
10688:  //Function trim( const s : string) : string
10689:  //Procedure setlength( var s : string; l : byte)
10690:  Function TimeZoneBias : longint
10691:  Function eight2seven_quoteprint( const s : string) : string
10692:  Function eight2seven_german( const s : string) : string
10693:  Function seven2eight_quoteprint( const s : string) : string end;
10694:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10695:  Function socketerror : cint
10696:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10697:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10698:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10699:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10700:  Function fplisten( s : cint; backlog : cint) : cint
10701:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10702:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10703:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10704:  Function NetAddrToStr( Entry : in_addr) : String
10705:  Function HostAddrToStr( Entry : in_addr) : String
10706:  Function StrToHostAddr( IP : String) : in_addr
10707:  Function StrToNetAddr( IP : String) : in_addr
10708:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10709:   cint8', 'shortint
10710:   cuint8', 'byte
10711:   cchar', 'cint8
10712:   cschar', 'cint8
10713:   cuchar', 'cuint8
10714:   cint16', 'smallint
10715:   cuint16', 'word
10716:   cshort', 'cint16
10717:   csshort', 'cint16
10718:   cushort', 'cuint16
10719:   cint32', 'longint
10720:   cuint32', 'longword
10721:   cint', 'cint32
10722:   csint', 'cint32
10723:   cuint', 'cuint32
10724:   csigned', 'cint
10725:   cunsigned', 'cuint
10726:   cint64', 'int64
10727:   clonglong', 'cint64
10728:   cslonglong', 'cint64
10729:   cbool', 'longbool
10730:   cfloat', 'single
10731:   cdouble', 'double
10732:   clongdouble', 'extended
10733:
10734: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10735: begin
10736:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10737:   SIRegister_TlkJSONdotnetclass(CL);
10738:   SIRegister_TlkJSONbase(CL);
10739:   SIRegister_TlkJSONnumber(CL);
10740:   SIRegister_TlkJSONstring(CL);
10741:   SIRegister_TlkJSONboolean(CL);
10742:   SIRegister_TlkJSONnull(CL);
10743:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10744:    +'se; data : TObject; var Continue : Boolean)
10745:   SIRegister_TlkJSONcustomlist(CL);
10746:   SIRegister_TlkJSONlist(CL);
10747:   SIRegister_TlkJSONobjectmethod(CL);
10748:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10749:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10750:   SIRegister_TlkHashTable(CL);
10751:   SIRegister_TlkBalTree(CL);
10752:   SIRegister_TlkJSONobject(CL);
10753:   SIRegister_TlkJSON(CL);
10754:   SIRegister_TlkJSONstreamed(CL);
10755:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10756: end;
10757:
10758: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10759: begin
10760:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10761:   SIRegister_TZSortedList(CL);
10762:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10763:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10764:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10765:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10766:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10767:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10768:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
```

```
10769:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10770:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10771:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10772:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10773:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10774:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10775:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10776:  Function StrToBoolEx( Str : string) : Boolean
10777:  Function BoolToStrEx( Bool : Boolean) : String
10778:  Function IsIpAddr( const Str : string) : Boolean
10779:  Function zSplitString( const Str, Delimiters : string) : TStrings
10780:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10781:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10782:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10783:  Function FloatToSQLStr( Value : Extended) : string
10784:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10785:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10786:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10787:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10788:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10789:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10790:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10791:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10792:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10793:  Function BytesToVar( const Value : TByteDynArray) : Variant
10794:  Function VarToBytes( const Value : Variant) : TByteDynArray
10795:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10796:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10797:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10798:  Function EncodeCString( const Value : string) : string
10799:  Function DecodeCString( const Value : string) : string
10800:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10801:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10802:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
        SubVersion:Int);
10803:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
        SubVersion:Integer):Int;
10804:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10805:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10806:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10807:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10808:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10809:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10810:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10811:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10812:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10813:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10814: end;
10815:
10816: unit uPSI_ZEncoding;
10817:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10818:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10819:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10820:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10821:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10822:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10823:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10824:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10825:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10826:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10827:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10828:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10829:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10830:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10831:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10832:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10833:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10834:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10835:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10836:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10837:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10838:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10839:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10840:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10841:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10842:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10843:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10844:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10845:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10846:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10847:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10848:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10849:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10850:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10851:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10852:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10853:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10854:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10855:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
```

```
10856:  Function ZDefaultSystemCodePage : Word
10857:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10858:
10859:
10860: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10861: begin
10862:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0);
10863:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1);
10864:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2);
10865:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3);
10866:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4);
10867:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5);
10868:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6);
10869:  {('alDefault','1 RPC_C_AUTHN_LEVEL_DEFAULT);
10870:  ('alNone','2 RPC_C_AUTHN_LEVEL_NONE);
10871:  ('alConnect','3 RPC_C_AUTHN_LEVEL_CONNECT);
10872:  ('alCall','4 RPC_C_AUTHN_LEVEL_CALL);
10873:  ('alPacket','5 RPC_C_AUTHN_LEVEL_PKT);
10874:  ('alPacketIntegrity','6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10875:  ('alPacketPrivacy','7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10876:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0);
10877:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1);
10878:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2);
10879:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3);
10880:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4);
10881:  {('ilDefault','0 RPC_C_IMP_LEVEL_DEFAULT);
10882:  ('ilAnonymous','1 RPC_C_IMP_LEVEL_ANONYMOUS);
10883:  ('ilIdentiry','2 RPC_C_IMP_LEVEL_IDENTIFY);
10884:  ('ilImpersonate','3 RPC_C_IMP_LEVEL_IMPERSONATE);
10885:  ('ilDelegate','4 RPC_C_IMP_LEVEL_DELEGATE);}
10886:  ('EOAC_NONE','LongWord').SetUInt( $0);
10887:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10888:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10889:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10890:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10891:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10892:  ('RPC_C_AUTHN_WINNT','LongInt'( 10);
10893:  ('RPC_C_AUTHNZ_NONE','LongInt'( 0);
10894:  ('RPC_C_AUTHNZ_NAME','LongInt'( 1);
10895:  ('RPC_C_AUTHNZ_DCE','LongInt'( 2);
10896:   FindClass('TOBJECT'),'EBoldCom
10897:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10898:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10899:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10900:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10901:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10902:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10903:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10904:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10905:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10906:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10907:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10908:  Function BoldCreateGUID : TGUID
10909:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
10910:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
10911:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10912:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10913: end;
10914:
10915: (*----------------------------------------------------------------------------*)
10916: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10917: begin
10918:  Function ParseISODate( s : string) : TDateTime
10919:  Function ParseISODateTime( s : string) : TDateTime
10920:  Function ParseISOTime( str : string) : TDateTime
10921: end;
10922:
10923: (*----------------------------------------------------------------------------*)
10924: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10925: begin
10926:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10927:  Function BoldCreateGUIDWithBracketsAsString : string
10928: end;
10929:
10930: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10931: begin
10932:   FindClass('TOBJECT'),'TBoldFileHandler
10933:   FindClass('TOBJECT'),'TBoldDiskFileHandler
10934:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
10935:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
10936:   SIRegister_TBoldFileHandler(CL);
10937:   SIRegister_TBoldDiskFileHandler(CL);
10938:  Procedure BoldCloseAllFilehandlers
10939:  Procedure BoldRemoveUnchangedFilesFromEditor
10940:  Function BoldFileHandlerList : TBoldObjectArray
10941:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
10942: end;
```

```
10943:
10944: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10945: begin
10946:   PCharArr', 'array of PChar
10947:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
10948:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
10949:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
10950:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
10951:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
10952:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
10953:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
10954:  Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
10955:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
10956:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
10957:  Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
       Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
10958:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
10959: end;
10960:
10961: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
10962: begin
10963:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
10964:   SIRegister_TfrmBoldQueryUser(CL);
10965:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
10966: end;
10967:
10968: (*----------------------------------------------------------------------------*)
10969: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
10970: begin
10971:  //('befIsInDisplayList',' BoldElementFlag0);
10972:  //('befStronglyDependedOfPrioritized',' BoldElementFlag1);
10973:  //('befFollowerSelected',' BoldElementFlag2);
10974:   FindClass('TOBJECT'),'TBoldQueue
10975:   FindClass('TOBJECT'),'TBoldQueueable
10976:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
10977:   SIRegister_TBoldQueueable(CL);
10978:   SIRegister_TBoldQueue(CL);
10979:  Function BoldQueueFinalized : Boolean
10980:  Function BoldInstalledQueue : TBoldQueue
10981: end;
10982:
10983: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
10984: begin
10985:  const mmPerInch','Extended').setExtended( 25.4);
10986:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
10987:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
10988:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
10989:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
10990:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
10991:   TBarLineType', '( white, black, black_half )
10992:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
10993:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
10994:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
10995:   TCheckSumMethod', '( csmNone, csmModulo10 )
10996:   SIRegister_TAsBarcode(CL);
10997:  Function CheckSumModulo10( const data : string) : string
10998:  Function ConvertMmToPixelsX( const Value : Double) : Integer
10999:  Function ConvertMmToPixelsY( const Value : Double) : Integer
11000:  Function ConvertInchToPixelsX( const Value : Double) : Integer
11001:  Function ConvertInchToPixelsY( const Value : Double) : Integer
11002: end;
11003:
11004: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
11005: begin
11006:    THomogeneousByteVector', 'array[0..3] of Byte
11007:   THomogeneousWordVector', 'array[0..3] of Word
11008:   THomogeneousIntVector', 'array[0..3] of Integer
11009:   THomogeneousFltVector', 'array[0..3] of single
11010:   THomogeneousDblVector', 'array[0..3] of double
11011:   THomogeneousExtVector', 'array[0..3] of extended
11012:   TAffineByteVector', 'array[0..2] of Byte
11013:   TAffineWordVector', 'array[0..2] of Word
11014:   TAffineIntVector', 'array[0..2] of Integer
11015:   TAffineFltVector', 'array[0..2] of single
11016:   TAffineDblVector', 'array[0..2] of double
11017:   TAffineExtVector', 'array[0..2] of extended
11018:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11019:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11020:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11021:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11022:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11023:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11024:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11025:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
```

```
11026:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11027:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11028:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11029:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11030:   TMatrix4b', 'THomogeneousByteMatrix
11031:   TMatrix4w', 'THomogeneousWordMatrix
11032:   TMatrix4i', 'THomogeneousIntMatrix
11033:   TMatrix4f', 'THomogeneousFltMatrix
11034:   TMatrix4d', 'THomogeneousDblMatrix
11035:   TMatrix4e', 'THomogeneousExtMatrix
11036:   TMatrix3b', 'TAffineByteMatrix
11037:   TMatrix3w', 'TAffineWordMatrix
11038:   TMatrix3i', 'TAffineIntMatrix
11039:   TMatrix3f', 'TAffineFltMatrix
11040:   TMatrix3d', 'TAffineDblMatrix
11041:   TMatrix3e', 'TAffineExtMatrix
11042:   //'PMatrix', '^TMatrix // will not work
11043:   TMatrixGL', 'THomogeneousFltMatrix
11044:   THomogeneousMatrix', 'THomogeneousFltMatrix
11045:   TAffineMatrix', 'TAffineFltMatrix
11046:   TQuaternion', 'record Vector : TVector4f; end
11047:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11048:   +'ger; Height : Integer; end
11049:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11050:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11051:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11052:  'EPSILON','Extended').setExtended( 1E-100);
11053:  'EPSILON2','Extended').setExtended( 1E-50);
11054:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11055:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11056:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11057:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11058:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11059:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11060:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11061:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11062:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11063:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11064:  Function VectorLength( V : array of Single) : Single
11065:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11066:  Procedure VectorNegate( V : array of Single)
11067:  Function VectorNorm( V : array of Single) : Single
11068:  Function VectorNormalize( V : array of Single) : Single
11069:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11070:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11071:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11072:  Procedure VectorScale( V : array of Single; Factor : Single)
11073:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11074:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11075:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11076:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11077:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11078:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11079:  Procedure MatrixAdjoint( var M : TMatrixGL)
11080:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11081:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11082:  Function MatrixDeterminant( M : TMatrixGL) : Single
11083:  Procedure MatrixInvert( var M : TMatrixGL)
11084:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11085:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11086:  Procedure MatrixTranspose( var M : TMatrixGL)
11087:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11088:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11089:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11090:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11091:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11092:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11093:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11094:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11095:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11096:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11097:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11098:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11099:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11100:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11101:  Function MakeAffineVector( V : array of Single) : TAffineVector
11102:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11103:  Function MakeVector( V : array of Single) : TVectorGL
11104:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11105:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11106:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11107:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11108:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11109:  Function ArcCosGL( X : Extended) : Extended
11110:  Function ArcSinGL( X : Extended) : Extended
11111:  Function ArcTan2GL( Y, X : Extended) : Extended
11112:  Function CoTanGL( X : Extended) : Extended
11113:  Function DegToRadGL( Degrees : Extended) : Extended
11114:  Function RadToDegGL( Radians : Extended) : Extended
```

```
11115:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11116:  Function TanGL( X : Extended) : Extended
11117:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11118:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11119:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11120:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11121:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11122:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11123: end;
11124:
11125:
11126: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11127: begin
11128:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11129:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11130:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11131:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11132:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11133:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11134:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11135:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11136:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11137:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11138:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11139:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11140:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11141:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11142:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11143:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11144:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11145:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11146:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11147:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11148:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11149:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11150:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11151:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11152:  Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11153:  Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
       SaveTo:TStrings):Bool;
11154:  Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11155: end;
11156:
11157: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11158: begin
11159:  CLSID_StdComponentCategoriesMgr','TGUID '{0002E005-0000-0000-C000-000000000046}
11160:  CATID_SafeForInitializing','TGUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11161:  CATID_SafeForScripting','TGUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11162:  icMAX_CATEGORY_DESC_LEN','LongInt'( 128);
11163:   FindClass('TOBJECT'),'EInvalidParam
11164:  Function IsDCOMInstalled : Boolean
11165:  Function IsDCOMEnabled : Boolean
11166:  Function GetDCOMVersion : string
11167:  Function GetMDACVersion : string
11168:  Function GetMDACVersion2 : string
11169:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11170:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11171:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11172:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11173:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11174:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11175:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11176:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11177:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11178:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11179:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11180:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11181:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11182:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11183: end;
11184:
11185:
11186: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11187: begin
11188:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11189:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11190:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11191:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11192:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11193:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11194:  DegPerCycle','Extended').setExtended( 360.0);
11195:  DegPerGrad','Extended').setExtended( 0.9);
11196:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11197:  GradPerCycle','Extended').setExtended( 400.0);
11198:  GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
```

```
11199:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11200:  RadPerCycle','Extended').setExtended( 6.283185307179586476925286766559);
11201:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11202:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11203:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11204:  CyclePerGrad','Extended').setExtended( 0.0025);
11205:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11206:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11207:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11208:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11209:  Function MakePercentage( const Step, Max : Longint) : Longint
11210:  Function CelsiusToKelvin( const T : double) : double
11211:  Function CelsiusToFahrenheit( const T : double) : double
11212:  Function KelvinToCelsius( const T : double) : double
11213:  Function KelvinToFahrenheit( const T : double) : double
11214:  Function FahrenheitToCelsius( const T : double) : double
11215:  Function FahrenheitToKelvin( const T : double) : double
11216:  Function CycleToDeg( const Cycles : double) : double
11217:  Function CycleToGrad( const Cycles : double) : double
11218:  Function CycleToRad( const Cycles : double) : double
11219:  Function DegToCycle( const Degrees : double) : double
11220:  Function DegToGrad( const Degrees : double) : double
11221:  Function DegToRad( const Degrees : double) : double
11222:  Function GradToCycle( const Grads : double) : double
11223:  Function GradToDeg( const Grads : double) : double
11224:  Function GradToRad( const Grads : double) : double
11225:  Function RadToCycle( const Radians : double) : double
11226:  Function RadToDeg( const Radians : double) : double
11227:  Function RadToGrad( const Radians : double) : double
11228:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11229:  Function DmsToRad( const D, M : Integer; const S : double) : double
11230:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11231:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11232:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11233:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11234:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11235:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11236:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11237:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11238:  Function CmToInch( const Cm : double) : double
11239:  Function InchToCm( const Inch : double) : double
11240:  Function FeetToMetre( const Feet : double) : double
11241:  Function MetreToFeet( const Metre : double) : double
11242:  Function YardToMetre( const Yard : double) : double
11243:  Function MetreToYard( const Metre : double) : double
11244:  Function NmToKm( const Nm : double) : double
11245:  Function KmToNm( const Km : double) : double
11246:  Function KmToSm( const Km : double) : double
11247:  Function SmToKm( const Sm : double) : double
11248:  Function LitreToGalUs( const Litre : double) : double
11249:  Function GalUsToLitre( const GalUs : double) : double
11250:  Function GalUsToGalCan( const GalUs : double) : double
11251:  Function GalCanToGalUs( const GalCan : double) : double
11252:  Function GalUsToGalUk( const GalUs : double) : double
11253:  Function GalUkToGalUs( const GalUk : double) : double
11254:  Function LitreToGalCan( const Litre : double) : double
11255:  Function GalCanToLitre( const GalCan : double) : double
11256:  Function LitreToGalUk( const Litre : double) : double
11257:  Function GalUkToLitre( const GalUk : double) : double
11258:  Function KgToLb( const Kg : double) : double
11259:  Function LbToKg( const Lb : double) : double
11260:  Function KgToOz( const Kg : double) : double
11261:  Function OzToKg( const Oz : double) : double
11262:  Function CwtUsToKg( const Cwt : double) : double
11263:  Function CwtUkToKg( const Cwt : double) : double
11264:  Function KaratToKg( const Karat : double) : double
11265:  Function KgToCwtUs( const Kg : double) : double
11266:  Function KgToCwtUk( const Kg : double) : double
11267:  Function KgToKarat( const Kg : double) : double
11268:  Function KgToSton( const Kg : double) : double
11269:  Function KgToLton( const Kg : double) : double
11270:  Function StonToKg( const STon : double) : double
11271:  Function LtonToKg( const Lton : double) : double
11272:  Function QrUsToKg( const Qr : double) : double
11273:  Function QrUkToKg( const Qr : double) : double
11274:  Function KgToQrUs( const Kg : double) : double
11275:  Function KgToQrUk( const Kg : double) : double
11276:  Function PascalToBar( const Pa : double) : double
11277:  Function PascalToAt( const Pa : double) : double
11278:  Function PascalToTorr( const Pa : double) : double
11279:  Function BarToPascal( const Bar : double) : double
11280:  Function AtToPascal( const At : double) : double
11281:  Function TorrToPascal( const Torr : double) : double
11282:  Function KnotToMs( const Knot : double) : double
11283:  Function HpElectricToWatt( const HpE : double) : double
11284:  Function HpMetricToWatt( const HpM : double) : double
11285:  Function MsToKnot( const ms : double) : double
11286:  Function WattToHpElectric( const W : double) : double
11287:  Function WattToHpMetric( const W : double) : double
```

```
11288:  function getBigPI: string;    //PI of 1000 numbers
11289:
11290: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11291: begin
11292:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11293:  Procedure CDCopyFile( const FileName, DestName : string)
11294:  Procedure CDMoveFile( const FileName, DestName : string)
11295:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11296:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11297:  Function CDGetTempDir : string
11298:  Function CDGetFileSize( FileName : string) : longint
11299:  Function GetFileTime( FileName : string) : longint
11300:  Function GetShortName( FileName : string) : string
11301:  Function GetFullName( FileName : string) : string
11302:  Function WinReboot : boolean
11303:  Function WinDir : String
11304:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11305:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11306:  Function devExecutor : TdevExecutor
11307: end;
11308:
11309: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11310: begin
11311:  Procedure CheckAssociations // AssociationsCount','LongInt'( 7);
11312:  Procedure Associate( Index : integer)
11313:  Procedure UnAssociate( Index : integer)
11314:  Function IsAssociated( Index : integer) : boolean
11315:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11316:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11317:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11318:  procedure RefreshIcons;
11319: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11320: function MergColor(Colors: Array of TColor): TColor;
11321: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11322: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11323: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11324: function GetInverseColor(AColor: TColor): TColor;
11325: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11326: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11327: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11328: Procedure GetSystemMenuFont(Font: TFont);
11329: end;
11330:
11331: //*************************unit uPSI_JvHLParser;*****************************
11332: function IsStringConstant(const St: string): Boolean;
11333: function IsIntConstant(const St: string): Boolean;
11334: function IsRealConstant(const St: string): Boolean;
11335: function IsIdentifier(const ID: string): Boolean;
11336: function GetStringValue(const St: string): string;
11337: procedure ParseString(const S: string; Ss: TStrings);
11338: function IsStringConstantW(const St: WideString): Boolean;
11339: function IsIntConstantW(const St: WideString): Boolean;
11340: function IsRealConstantW(const St: WideString): Boolean;
11341: function IsIdentifierW(const ID: WideString): Boolean;
11342: function GetStringValueW(const St: WideString): WideString;
11343: procedure ParseStringW(const S: WideString; Ss: TStrings);
11344:
11345:
11346: //*************************unit uPSI_JclMapi;*****************************
11347:
11348: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11349: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11350: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool
11351: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11352: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11353:
11354: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11355: begin
11356:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11357:  Function BuildType1Message( ADomain, AHost : String) : String
11358:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11359:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11360:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11361: GBase64CodeTable','string'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11362: GXXECodeTable','string'+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11363: GUUECodeTable','string'`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11364: end;
11365:
11366: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11367: begin
11368: ('IpAny','LongWord').SetUInt( $00000000);
11369:  IpLoopBack','LongWord').SetUInt( $7F000001);
11370:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11371:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11372:  PortAny','LongWord( $0000);
11373:  SocketMaxConnections','LongInt'( 5);
```

```
11374:  TIpAddr', 'LongWord
11375:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11376:  Function HostToNetLong( HostLong : LongWord) : LongWord
11377:  Function HostToNetShort( HostShort : Word) : Word
11378:  Function NetToHostLong( NetLong : LongWord) : LongWord
11379:  Function NetToHostShort( NetShort : Word) : Word
11380:  Function StrToIp( Ip : string) : TIpAddr
11381:  Function IpToStr( Ip : TIpAddr) : string
11382: end;
11383:
11384: (*----------------------------------------------------------------------------*)
11385: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11386: begin
11387:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11388:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11389:    +'amSha1, AlsmtpClientAuthAutoSelect )
11390:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11391:   SIRegister_TAlSmtpClient(CL);
11392: end;
11393:
11394: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11395: begin
11396: 'TBitNo', 'Integer
11397:   TStByteNo', 'Integer
11398: TStationNo', 'Integer
11399: TInOutNo', 'Integer
11400: TIo', '( EE, AA, NE, NA )
11401: TBitSet', 'set of TBitNo
11402: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11403: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11404: TBitAddr', 'LongInt
11405: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11406: TByteAddr', 'SmallInt
11407: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11408:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11409:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11410:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
      aBitNo:TBitNo);
11411:  Function BitAddrToStr( Value : TBitAddr) : string
11412:  Function StrToBitAddr( const Value : string) : TBitAddr
11413:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11414:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11415:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11416:  Function ByteAddrToStr( Value : TByteAddr) : string
11417:  Function StrToByteAddr( const Value : string) : TByteAddr
11418:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11419:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11420:  Function InOutStateToStr( State : TInOutState) : string
11421:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11422:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11423: end;
11424:
11425: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11426: begin
11427: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11428:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11429: DpmiPmVector', 'Int64
11430:  'DInterval','LongInt'( 1000);
11431:  //'DEnabled','Boolean')BoolToStr( True);
11432:  'DIntFreq','string' if64
11433:  //'DMessages','Boolean if64);
11434:   SIRegister_TwdxCustomTimer(CL);
11435:   SIRegister_TwdxTimer(CL);
11436:   SIRegister_TwdxRtcTimer(CL);
11437:   SIRegister_TCustomIntTimer(CL);
11438:   SIRegister_TIntTimer(CL);
11439:   SIRegister_TRtcIntTimer(CL);
11440:  Function RealNow : TDateTime
11441:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11442:  Function DateTimeToMs( Time : TDateTime) : LongInt
11443: end;
11444:
11445: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11446: begin
11447: TIdSyslogPRI', 'Integer
11448: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11449:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11450:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11451:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11452:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11453: TIdSyslogSeverity','(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11454:   SIRegister_TIdSysLogMsgPart(CL);
11455:   SIRegister_TIdSysLogMessage(CL);
11456:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11457:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11458:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11459:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11460:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11461:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
```

```
11462: end;
11463:
11464: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11465: begin
11466:  'UWhitespace','String '(?:\s*)
11467:  Function StripSpaces( const AText : string) : string
11468:  Function CharCount( const AText : string; Ch : Char) : Integer
11469:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11470:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11471: end;
11472:
11473:
11474: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11475: begin
11476:  ExtPascalVersion','String '0.9.8
11477:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11478:    +'Opera, brKonqueror, brMobileSafari )
11479:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11480:   AddTypeS('TExtProcedure', 'Procedure
11481:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11482:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11483:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11484:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11485:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11486:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11487:  Function StrToJS( const S : string; UseBR : boolean) : string
11488:  Function CaseOf( const S : string; const Cases : array of string) : integer
11489:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11490:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11491:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11492:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11493:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11494:  Function IsUpperCase( S : string) : boolean
11495:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11496:  Function BeautifyCSS( const AStyle : string) : string
11497:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11498:  Function JSDateToDateTime( JSDate : string) : TDateTime
11499: end;
11500:
11501: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11502: begin
11503:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11504:   TSHDeleteOptions', 'set of TSHDeleteOption
11505:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11506:   TSHRenameOptions', 'set of TSHRenameOption
11507:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11508:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11509:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11510:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11511:   TEnumFolderFlags', 'set of TEnumFolderFlag
11512:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11513:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11514:    +'IEnumIdList; Folder : IShellFolder; end
11515:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11516:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11517:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11518:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11519:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11520:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11521:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11522:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11523:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11524:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11525:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11526:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11527:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11528:  Function SHFreeMem( var P : Pointer) : Boolean
11529:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11530:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11531:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11532:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11533:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11534:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11535:  Function PidlFree( var IdList : PItemIdList) : Boolean
11536:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11537:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11538:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11539:  Function PidlToPath( IdList : PItemIdList) : string
11540:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11541:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11542:   PShellLink', '^TShellLink // will not work
11543:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11544:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11545:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11546:  Procedure ShellLinkFree( var Link : TShellLink)
11547:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11548:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11549:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11550:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
```

```
11551:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11552:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11553:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11554:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11555:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11556:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11557:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11558:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11559:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11560:  Function ShellOpenAs( const FileName : string) : Boolean
11561:  Function ShellRasDial( const EntryName : string) : Boolean
11562:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11563:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11564:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11565:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11566:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11567:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11568:  Function OemKeyScan( wOemChar : Word) : DWORD
11569:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11570: end;
11571:
11572: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11573: begin
11574:  xmlVersion','String '1.0 FindClass('TOBJECT'),'Exml
11575:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11576:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11577:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11578:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11579:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11580:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11581:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11582:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11583:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11584:  Function xmlValidName( const Text : UnicodeString) : Boolean
11585:  //xmlSpace','Char #$20 or  #$9 or  #$D or  #$A);
11586:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11587:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11588:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11589:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11590:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11591:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11592:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11593:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11594:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11595:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11596:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11597:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11598:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11599:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11600:  Procedure SelfTestcXMLFunctions
11601: end;
11602:
11603: (*----------------------------------------------------------------------------*)
11604: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11605: begin
11606:  Function AWaitCursor : IUnknown
11607:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11608:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11609:  Function YesNo( const ACaption, AMsg : string) : boolean
11610:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11611:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11612:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11613:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11614:  Procedure GetSystemPaths( Strings : TStrings)
11615:  Procedure MakeEditNumeric( EditHandle : integer)
11616: end;
11617:
11618: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11619: begin
11620:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11621:  'BI_YUY2','LongWord( $32595559);
11622:  'BI_UYVY','LongWord').SetUInt( $59565955);
11623:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11624:  'BI_YVU9','LongWord').SetUInt( $39555659);
11625:  'BI_YUV12','LongWord( $30323449);
11626:  'BI_Y8','LongWord').SetUInt( $20203859);
11627:  'BI_Y211','LongWord').SetUInt( $31313259);
11628:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11629:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11630: end;
11631:
11632: (*----------------------------------------------------------------------------*)
11633: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11634: begin
11635:  'WM_USER','LongWord').SetUInt( $0400);
11636:  'WM_CAP_START','LongWord').SetUint($0400);
11637:  'WM_CAP_END','longword').SetUint($0400+85);
11638:  //WM_CAP_START+  85
```

```
11639:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11640:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11641:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11642:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11643:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11644:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11645:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11646:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11647:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11648:  Function capGetUserData( hwnd : THandle) : LongInt
11649:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11650:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11651:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11652:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11653:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11654:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11655:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11656:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11657:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11658:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11659:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11660:  Function capEditCopy( hwnd : THandle) : LongInt
11661:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11662:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11663:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11664:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11665:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11666:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11667:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11668:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11669:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11670:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11671:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11672:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11673:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11674:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11675:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11676:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11677:  Function capGrabFrame( hwnd : THandle) : LongInt
11678:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11679:  Function capCaptureSequence( hwnd : THandle) : LongInt
11680:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11681:  Function capCaptureStop( hwnd : THandle) : LongInt
11682:  Function capCaptureAbort( hwnd : THandle) : LongInt
11683:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11684:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11685:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11686:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11687:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11688:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11689:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11690:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11691:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11692:  Function capPalettePaste( hwnd : THandle) : LongInt
11693:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11694:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11695:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11696:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11697:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11698:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11699:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11700:   //PCapStatus', '^TCapStatus // will not work
11701:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11702:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11703:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11704:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11705:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11706:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11707:    +' wNumAudioAllocated : WORD; end
11708:   //PCaptureParms', '^TCaptureParms // will not work
11709:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11710:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11711:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11712:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11713:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11714:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11715:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11716:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11717:    +'he : BOOL; AVStreamMaster : WORD; end
11718:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11719:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11720:  'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11721:  'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11722:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
      : Integer; hwndParent : THandle; nID : Integer) : THandle
11723:  Function
     capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11724:  'IDS_CAP_BEGIN','LongInt'( 300);
11725:  'IDS_CAP_END','LongInt'( 301);
```

```
11726:  'IDS_CAP_INFO','LongInt'( 401);
11727:  'IDS_CAP_OUTOFMEM','LongInt'( 402);
11728:  'IDS_CAP_FILEEXISTS','LongInt'( 403);
11729:  'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11730:  'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11731:  'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11732:  'IDS_CAP_DEFAVIEXT','LongInt'( 407);
11733:  'IDS_CAP_DEFPALEXT','LongInt'( 408);
11734:  'IDS_CAP_CANTOPEN','LongInt'( 409);
11735:  'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11736:  'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11737:  'IDS_CAP_VIDEDITERR','LongInt'( 412);
11738:  'IDS_CAP_READONLYFILE','LongInt'( 413);
11739:  'IDS_CAP_WRITEERROR','LongInt'( 414);
11740:  'IDS_CAP_NODISKSPACE','LongInt'( 415);
11741:  'IDS_CAP_SETFILESIZE','LongInt'( 416);
11742:  'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11743:  'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11744:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11745:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11746:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11747:  'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11748:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11749:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11750:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11751:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11752:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11753:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11754:  'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11755:  'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11756:  'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11757:  'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11758:  'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11759:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11760:  'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11761:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11762:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11763:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11764:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11765:  'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11766:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11767:  'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11768:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11769:  'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11770:  'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11771:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11772:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11773:  'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11774:  'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11775:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11776:  'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11777:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11778:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11779:  'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11780:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11781:  'AVICAP32','String 'AVICAP32.dll
11782: end;
11783:
11784: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11785: begin
11786:  Function AlBoolToInt( Value : Boolean) : Integer
11787:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11788:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11789:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11790:  Function ALInc( var x : integer; Count : integer) : Integer
11791:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11792:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11793:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11794:  Function ALIsInteger(const S: AnsiString): Boolean;
11795:  function ALIsDecimal(const S: AnsiString): boolean;
11796:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11797:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11798:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11799:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11800:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11801:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11802:  Function ALRandomStr(const aLength: Longint): AnsiString;
11803:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11804:  Function ALRandomStrU(const aLength: Longint): String;
11805: end;
11806:
11807: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11808: begin
11809:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
      aTrueStr: AnsiString; const aFalseStr : AnsiString)
11810: end;
11811:
11812: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11813: begin
```

```
11814:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11815:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11816:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11817:    +'; ullAvailExtendedVirtual : Int64; end
11818:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11819:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11820:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11821:  'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ));
11822:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11823:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11824:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11825:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11826: end;
11827:
11828: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11829: begin
11830:   SIRegister_THandledObject(CL);
11831:   SIRegister_TEvent(CL);
11832:   SIRegister_TMutex(CL);
11833:   SIRegister_TSharedMem(CL);
11834:  'TRACE_BUF_SIZE','LongInt'( 200 * 1024);
11835:  'TRACE_BUFFER','String 'TRACE_BUFFER
11836:  'TRACE_MUTEX','String 'TRACE_MUTEX
11837:   //PTraceEntry', '^TTraceEntry // will not work
11838:   SIRegister_TIPCTracer(CL);
11839:  'MAX_CLIENTS','LongInt'( 6);
11840:  'IPCTIMEOUT','LongInt'( 2000);
11841:  'IPCBUFFER_NAME','String 'BUFFER_NAME
11842:  'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11843:  'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11844:  'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11845:  'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11846:  'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11847:  'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11848:   FindClass('TOBJECT'),'EMonitorActive
11849:   FindClass('TOBJECT'),'TIPCThread
11850:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11851:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11852:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11853:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11854:   TClientFlags', 'set of TClientFlag
11855:   //PEventData', '^TEventData // will not work
11856:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11857:    +'lag; Flags : TClientFlags; end
11858:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11859:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11860:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11861:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11862:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11863:   SIRegister_TIPCEvent(CL);
11864:   //PClientDirRecords', '^TClientDirRecords // will not work
11865:   SIRegister_TClientDirectory(CL);
11866:   TIPCState', '( stInActive, stDisconnected, stConnected )
11867:   SIRegister_TIPCThread(CL);
11868:   SIRegister_TIPCMonitor(CL);
11869:   SIRegister_TIPCClient(CL);
11870:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11871: end;
11872:
11873: (*----------------------------------------------------------------------------*)
11874: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11875: begin
11876:   SIRegister_TAlGSMComm(CL);
11877:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11878:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
      AMessage:AnsiString);
11879:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11880:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
      UseGreekAlphabet:Bool):Widestring;
11881:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11882:  end;
11883:
11884: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11885: begin
11886:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11887:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11888:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11889:   TInternetScheme', 'integer
11890:   TALIPv6Binary', 'array[1..16] of Char;
11891:  // TALIPv6Binary = array[1..16] of ansiChar;
11892:  //   TInternetScheme = Integer;
11893:   SIRegister_TALHTTPRequestHeader(CL);
11894:   SIRegister_TALHTTPCookie(CL);
11895:   SIRegister_TALHTTPCookieCollection(CL);
11896:   SIRegister_TALHTTPResponseHeader(CL);
11897:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11898:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11899: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
      Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
```

```
11900: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
       Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11901: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
       Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11902:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11903:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11904:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11905:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11906:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11907:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
       ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11908:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
       Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11909:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11910:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11911:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11912:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11913:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11914:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11915:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11916:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11917:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11918:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11919:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
11920:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11921:  Function ALZeroIpV6 : TALIPv6Binary
11922:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11923:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
11924:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11925:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
11926: end;
11927:
11928: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11929: begin
11930:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Bool;
11931:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11932:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
11933:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
11934:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
11935:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
11936:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
11937:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
11938:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
11939:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11940:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
11941: end;
11942:
11943: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11944: begin
11945:   SIRegister_TALEMailHeader(CL);
11946:   SIRegister_TALNewsArticleHeader(CL);
11947:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
11948:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
11949:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
11950:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
11951:  Function AlGenerateInternetMessageID : AnsiString;
11952:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
11953:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
11954:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11955: end;
11956:
11957: (*----------------------------------------------------------------------------*)
11958: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
11959: begin
11960:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11961:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
11962:  Function ALgetLocalIPs : TALStrings
11963:  Function ALgetLocalHostName : AnsiString
11964: end;
11965:
11966: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
11967: begin
11968:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
11969:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11970:  Procedure AlCGIInitDefaultServerVariables( ServerVariables : TALStrings);
11971:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiStr;
11972:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11973:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
11974:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
```

```
11975:  +'overloadedRequestContentStream:Tstream;var
        ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
11976:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
        InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
        ResponseHeader : TALHTTPResponseHeader);
11977: end;
11978:
11979: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
11980: begin
11981:   TStartupInfoA', 'TStartupInfo
11982:  'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
11983:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege
11984:  SE_LOCK_MEMORY_NAME','String)( 'SeLockMemoryPrivilege
11985:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege
11986:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege
11987:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege
11988:  SE_TCB_NAME','String 'SeTcbPrivilege
11989:  SE_SECURITY_NAME','String 'SeSecurityPrivilege
11990:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege
11991:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege
11992:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege
11993:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege
11994:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege
11995:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege
11996:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege
11997:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege
11998:  SE_BACKUP_NAME','String 'SeBackupPrivilege
11999:  SE_RESTORE_NAME','String 'SeRestorePrivilege
12000:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege
12001:  SE_DEBUG_NAME','String 'SeDebugPrivilege
12002:  SE_AUDIT_NAME','String 'SeAuditPrivilege
12003:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege
12004:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege
12005:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege
12006:  SE_UNDOCK_NAME','String 'SeUndockPrivilege
12007:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege
12008:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege
12009:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege
12010:  Function AlGetEnvironmentString : AnsiString
12011:  Function ALWinExec32(const FileName,CurrentDir,
        Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12012:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12013:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12014:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12015:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12016: end;
12017:
12018: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12019: begin
12020:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
        RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12021:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
        RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12022:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
        FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12023:  Function ALGetModuleName : ansistring
12024:  Function ALGetModuleFileNameWithoutExtension : ansistring
12025:  Function ALGetModulePath : ansiString
12026:  Function AlGetFileSize( const AFileName : ansistring) : int64
12027:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
12028:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
12029:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12030:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12031:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12032:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12033:  Function ALFileExists( const Path : ansiString) : boolean
12034:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12035:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12036:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12037:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12038:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12039: end;
12040:
12041: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12042: begin
12043:   NativeInt', 'Integer
12044:   NativeUInt', 'Cardinal
12045:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12046:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12047:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12048:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12049:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12050:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12051:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
        InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12052:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
        InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12053:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
        InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
```

```
12054:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12055:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12056:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12057:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12058:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12059:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12060:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12061:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12062:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12063:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
12064:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12065:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12066:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12067:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12068:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12069:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12070:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76);
12071:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12072:  'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12073:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12074:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12075:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12076:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12077: end;
12078:
12079: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12080: begin
12081:  'cALXMLNodeMaxListSize','LongInt'( Maxint div 16);
12082:   FindClass('TOBJECT'),'TALXMLNode
12083:   FindClass('TOBJECT'),'TALXMLNodeList
12084:   FindClass('TOBJECT'),'TALXMLDocument
12085:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12086:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12087:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12088:    +'nst Name : AnsiString; const Attributes : TALStrings)
12089:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12090:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12091:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12092:    +'ntDocType, ntDocFragment, ntNotation )
12093:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12094:   TALXMLDocOptions', 'set of TALXMLDocOption
12095:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12096:   TALXMLParseOptions', 'set of TALXMLParseOption
12097:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12098:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12099:   SIRegister_EALXMLDocError(CL);
12100:   SIRegister_TALXMLNodeList(CL);
12101:   SIRegister_TALXMLNode(CL);
12102:   SIRegister_TALXmlElementNode(CL);
12103:   SIRegister_TALXmlAttributeNode(CL);
12104:   SIRegister_TALXmlTextNode(CL);
12105:   SIRegister_TALXmlDocumentNode(CL);
12106:   SIRegister_TALXmlCommentNode(CL);
12107:   SIRegister_TALXmlProcessingInstrNode(CL);
12108:   SIRegister_TALXmlCDataNode(CL);
12109:   SIRegister_TALXmlEntityRefNode(CL);
12110:   SIRegister_TALXmlEntityNode(CL);
12111:   SIRegister_TALXmlDocTypeNode(CL);
12112:   SIRegister_TALXmlDocFragmentNode(CL);
12113:   SIRegister_TALXmlNotationNode(CL);
12114:   SIRegister_TALXMLDocument(CL);
12115:  cAlXMLUTF8EncodingStr','String 'UTF-8
12116:  cALXmlUTF8HeaderStr','String'<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12117:  CALNSDelim','String ':
12118:  CALXML','String 'xml
12119:  CALVersion','String 'version
12120:  CALEncoding','String 'encoding
12121:  CALStandalone','String 'standalone
12122:  CALDefaultNodeIndent','String '
12123:  CALXmlDocument','String 'DOCUMENT
12124:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12125:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
       EncodingStr:AnsiString);
12126:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
       ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12127:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
       ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12128:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
       Recurse: Boolean):TalxmlNode
12129:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
       AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
```

```
12130:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
       AnsiString
12131: end;
12132:
12133: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12134: //based on TEEProc, TeCanvas, TEEngine, TChart
12135: begin
12136:  'TeePiStep','Double').setExtended( Pi / 180.0);
12137:  'TeeDefaultPerspective','LongInt'( 100);
12138:  'TeeMinAngle','LongInt'( 270);
12139:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12140:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12141:  'teeclCream','LongWord( TColor ( $F0FBFF ));
12142:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12143:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12144:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12145:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12146:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12147:  'TA_LEFT','LongInt'( 0);
12148:  'TA_RIGHT','LongInt'( 2);
12149:  'TA_CENTER','LongInt'( 6);
12150:  'TA_TOP','LongInt'( 0);
12151:  'TA_BOTTOM','LongInt'( 8);
12152:  'teePATCOPY','LongInt'( 0);
12153:  'NumCirclePoints','LongInt'( 64);
12154:  'teeDEFAULT_CHARSET','LongInt'( 1);
12155:  'teeANTIALIASED_QUALITY','LongInt'( 4);
12156:  'TA_LEFT','LongInt'( 0);
12157:  'bs_Solid','LongInt'( 0);
12158:  'teepf24Bit','LongInt'( 0);
12159:  'teepfDevice','LongInt'( 1);
12160:  'CM_MOUSELEAVE','LongInt'( 10000);
12161:  'CM_SYSCOLORCHANGE','LongInt'( 10001);
12162:  'DC_BRUSH','LongInt'( 18);
12163:  'DC_PEN','LongInt'( 19);
12164:   teeCOLORREF', 'LongWord
12165:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12166:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12167:   SIRegister_TFilterRegion(CL);
12168:   SIRegister_IFormCreator(CL);
12169:   SIRegister_TTeeFilter(CL);
12170:   //TFilterClass', 'class of TTeeFilter
12171:   SIRegister_TFilterItems(CL);
12172:   SIRegister_TConvolveFilter(CL);
12173:   SIRegister_TBlurFilter(CL);
12174:   SIRegister_TTeePicture(CL);
12175:   TPenEndStyle', '( esRound, esSquare, esFlat )
12176:   SIRegister_TChartPen(CL);
12177:   SIRegister_TChartHiddenPen(CL);
12178:   SIRegister_TDottedGrayPen(CL);
12179:   SIRegister_TDarkGrayPen(CL);
12180:   SIRegister_TWhitePen(CL);
12181:   SIRegister_TChartBrush(CL);
12182:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12183:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12184:   SIRegister_TView3DOptions(CL);
12185:   FindClass('TOBJECT'),'TTeeCanvas
12186:   TTeeTransparency', 'Integer
12187:   SIRegister_TTeeBlend(CL);
12188:   FindClass('TOBJECT'),'TCanvas3D
12189:   SIRegister_TTeeShadow(CL);
12190:   teeTGradientDirection','(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
       gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12191:   FindClass('TOBJECT'),'TSubGradient
12192:   SIRegister_TCustomTeeGradient(CL);
12193:   SIRegister_TSubGradient(CL);
12194:   SIRegister_TTeeGradient(CL);
12195:   SIRegister_TTeeFontGradient(CL);
12196:   SIRegister_TTeeFont(CL);
12197:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12198:   TCanvasTextAlign', 'Integer
12199:   TTeeCanvasHandle', 'HDC
12200:   SIRegister_TTeeCanvas(CL);
12201:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12202:   SIRegister_TFloatXYZ(CL);
12203:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12204:   TRGB', 'record blue: byte; green: byte; red: byte; end
12205:   {TRGB=packed record
12206:     Blue  : Byte;
12207:     Green : Byte;
12208:     Red   : Byte;
12209:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12210:
12211: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12212:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12213: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12214: TCanvas3DPlane', '( cpX, cpY, cpZ )
12215:   //IInterface', 'IUnknown
12216:   SIRegister_TCanvas3D(CL);
```

```
12217:   SIRegister_TTeeCanvas3D(CL);
12218:   TTrianglePoints', 'Array[0..2] of TPoint;
12219:   TFourPoints', 'Array[0..3] of TPoint;
12220:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12221:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12222:  Function Point3D( const x, y, z : Integer) : TPoint3D
12223:  Procedure SwapDouble( var a, b : Double)
12224:  Procedure SwapInteger( var a, b : Integer)
12225:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12226:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12227:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12228:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12229:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12230:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12231:  Procedure UnClipCanvas( ACanvas : TCanvas)
12232:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12233:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12234:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12235:  'TeeCharForHeight','String 'W
12236:  'DarkerColorQuantity','Byte').SetUInt( 128);
12237:  'DarkColorQuantity','Byte').SetUInt( 64);
12238:   TButtonGetColorProc', 'Function  : TColor
12239:   SIRegister_TTeeButton(CL);
12240:   SIRegister_TButtonColor(CL);
12241:   SIRegister_TComboFlat(CL);
12242:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12243:  Function TeePoint( const aX, aY : Integer) : TPoint
12244:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12245:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12246:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12247:  Function OrientRectangle( const R : TRect) : TRect
12248:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12249:  Function PolygonBounds( const P : array of TPoint) : TRect
12250:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12251:  Function RGBValue( const Color : TColor) : TRGB
12252:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12253:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12254:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12255:  Function TeeCull( const P : TFourPoints) : Boolean;
12256:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12257:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12258:  Procedure SmoothStretch( Src, Dst : TBitmap);
12259:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12260:  Function TeeDistance( const x, y : Double) : Double
12261:  Function TeeLoadLibrary( const FileName : String) : HInst
12262:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12263:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12264:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12265:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
      Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12266:   SIRegister_ICanvasHyperlinks(CL);
12267:   SIRegister_ICanvasToolTips(CL);
12268:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12269: end;
12270:
12271: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12272: begin
12273:   TOvcHdc', 'Integer
12274:   TOvcHWND', 'Cardinal
12275:   TOvcHdc', 'HDC
12276:   TOvcHWND', 'HWND
12277:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12278:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12279:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12280:  Function DefaultEpoch : Integer
12281:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12282:  Procedure FixRealPrim( P : PChar; DC : Char)
12283:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12284:  Function GetLeftButton : Byte
12285:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12286:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12287:  Function GetShiftFlags : Byte
12288:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12289:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12290:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12291:  Function ovIsForegroundTask : Boolean
12292:  Function ovTrimLeft( const S : string) : string
12293:  Function ovTrimRight( const S : string) : string
12294:  Function ovQuotedStr( const S : string) : string
12295:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12296:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12297:  Function PtrDiff( const P1, P2 : PChar) : Word
12298:  Procedure PtrInc( var P, Delta : Word)
12299:  Procedure PtrDec( var P, Delta : Word)
12300:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12301:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
      SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12302:  Function ovMinI( X, Y : Integer) : Integer
12303:  Function ovMaxI( X, Y : Integer) : Integer
```

```
12304:  Function ovMinL( X, Y : LongInt) : LongInt
12305:  Function ovMaxL( X, Y : LongInt) : LongInt
12306:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12307:  Function PartialCompare( const S1, S2 : string) : Boolean
12308:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12309:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12310:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12311:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
       TransparentColor : TColor)
12312:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
       TransparentColor : TColorRef)
12313:  Function ovWidthOf( const R : TRect) : Integer
12314:  Function ovHeightOf( const R : TRect) : Integer
12315:  Procedure ovDebugOutput( const S : string)
12316:  Function GetArrowWidth( Width, Height : Integer) : Integer
12317:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12318:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12319:  Procedure StripCharFromFront( aChr : Char; var Str : string)
12320:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12321:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12322:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12323:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12324:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12325:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12326:  Function CreateMetaFile( p1 : PChar) : HDC
12327:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12328:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12329:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12330:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12331:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12332:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12333:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12334:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12335:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12336:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12337:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12338:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12339:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
       SrcHeight:Int;Rop:DWORD):BOOL
12340:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12341:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
       SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12342:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12343:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12344:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12345:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12346:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12347:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12348:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12349:  Function UpdateColors( DC : HDC) : BOOL
12350:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12351:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12352:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12353:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12354:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12355:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12356:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Integer; SrcDC : HDC; XScr, YScr : Integer; Mask :
       HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12357:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12358:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12359:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12360:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12361:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12362:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12363:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12364:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12365:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12366:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12367:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12368:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12369:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12370: end;
12371:
12372: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12373: begin
12374:   SIRegister_TOvcAbstractStore(CL);
12375:   //PExPropInfo', '^TExPropInfo // will not work
12376: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12377:   SIRegister_TOvcPropertyList(CL);
12378:   SIRegister_TOvcDataFiler(CL);
12379:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12380:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12381:  Function CreateStoredItem( const CompName, PropName : string) : string
12382:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12383:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12384: end;
12385:
12386: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
```

```
12387: begin
12388:  'ovsetsize','LongInt'( 16);
12389:  'etSyntax','LongInt'( 0);
12390:  'etSymantic','LongInt'( 1);
12391:  'chCR','Char #13);
12392:  'chLF','Char #10);
12393:  'chLineSeparator',' chCR);
12394:   SIRegister_TCocoError(CL);
12395:   SIRegister_TCommentItem(CL);
12396:   SIRegister_TCommentList(CL);
12397:   SIRegister_TSymbolPosition(CL);
12398: TGenListType', '( glNever, glAlways, glOnError )
12399: TovBitSet', 'set of Integer
12400:   //PStartTable', '^TStartTable // will not work
12401: 'TovCharSet', 'set of AnsiChar
12402: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12403: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12404: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12405: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12406: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12407:    +'osition; const Data : string; ErrorType : integer)
12408: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12409: TGetCH', 'Function ( pos : longint) : char
12410: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12411:   SIRegister_TCocoRScanner(CL);
12412:   SIRegister_TCocoRGrammar(CL);
12413:  '_EF','Char #0);
12414:  '_TAB','Char').SetString( #09);
12415:  '_CR','Char').SetString( #13);
12416:  '_LF','Char').SetString( #10);
12417:  '_EL','').SetString( _CR);
12418:  '_EOF','Char').SetString( #26);
12419:  'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12420:  'minErrDist','LongInt'( 2);
12421:  Function ovPadL( S : string; ch : char; L : integer) : string
12422: end;
12423:
12424:   TFormatSettings = record
12425:     CurrencyFormat: Byte;
12426:     NegCurrFormat: Byte;
12427:     ThousandSeparator: Char;
12428:     DecimalSeparator: Char;
12429:     CurrencyDecimals: Byte;
12430:     DateSeparator: Char;
12431:     TimeSeparator: Char;
12432:     ListSeparator: Char;
12433:     CurrencyString: string;
12434:     ShortDateFormat: string;
12435:     LongDateFormat: string;
12436:     TimeAMString: string;
12437:     TimePMString: string;
12438:     ShortTimeFormat: string;
12439:     LongTimeFormat: string;
12440:     ShortMonthNames: array[1..12] of string;
12441:     LongMonthNames: array[1..12] of string;
12442:     ShortDayNames: array[1..7] of string;
12443:     LongDayNames: array[1..7] of string;
12444:     TwoDigitYearCenturyWindow: Word;
12445:    end;
12446:
12447: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12448: begin
12449:  Function ovFormatSettings : TFormatSettings
12450: end;
12451:
12452: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12453: begin
12454:   TOvcCharSet', 'set of Char
12455:   ovBTable', 'array[0..255] of Byte
12456:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
      HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12457:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12458:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12459:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12460:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12461:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12462:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12463:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12464:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12465:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12466:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12467:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12468:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12469:  Function LoCaseChar( C : Char) : Char
12470:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12471:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12472:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12473:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12474:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
```

```
12475:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12476:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12477:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12478:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12479:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12480:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12481:  Procedure TrimAllSpacesPChar( P : PChar)
12482:  Function TrimEmbeddedZeros( const S : string) : string
12483:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12484:  Function TrimTrailPrimPChar( S : PChar) : PChar
12485:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12486:  Function TrimTrailingZeros( const S : string) : string
12487:  Procedure TrimTrailingZerosPChar( P : PChar)
12488:  Function UpCaseChar( C : Char) : Char
12489:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12490:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12491:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12492: end;
12493:
12494: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12495: begin
12496:   //PRaiseFrame', '^TRaiseFrame // will not work
12497:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12498:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12499:  Procedure SafeCloseHandle( var Handle : THandle)
12500:  Procedure ExchangeInteger( X1, X2 : Integer)
12501:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12502:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12503:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12504:
12505: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12506:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12507: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12508:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12509:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12510:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12511:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12512:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12513:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12514:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12515:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12516:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12517:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12518:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12519:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12520:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12521:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12522:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12523:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12524:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12525:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12526:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12527:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12528:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12529:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12530:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12531:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12532:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12533:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12534:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12535:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12536:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12537:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12538:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12539:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12540:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12541:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12542:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12543:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12544:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12545:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12546:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12547:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12548:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12549:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12550:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12551:       var lpLuid: TLargeInteger): BOOL; stdcall;
12552:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12553:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12554:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12555:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12556:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12557:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12558:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12559:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12560:       var GenerateOnClose: BOOL): BOOL; stdcall;
12561:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12562:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12563:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
```

```
12564:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12565:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12566:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12567:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12568:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12569:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12570:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12571:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12572:       var phkResult: HKEY): Longint; stdcall;
12573:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12574:       var phkResult: HKEY): Longint; stdcall;
12575:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12576:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12577:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12578:       lpdwDisposition: PDWORD): Longint; stdcall;
12579:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12580:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12581:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12582:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12583:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12584:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12585:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12586:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12587:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12588:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12589:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12590:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12591:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12592:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12593:       lpcbClass: PDWORD; lpReserved: Pointer;
12594:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12595:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12596:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12597:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12598:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12599:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12600:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12601:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12602:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12603:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12604:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12605:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12606:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12607:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12608:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12609:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12610:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12611:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12612:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12613:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12614:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12615:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12616:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12617:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12618:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12619:
12620:  Function wAddAtom( lpString : PKOLChar) : ATOM
12621:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12622:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12623:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12624:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
      lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12625:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12626:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12627:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12628:  Function wCreateDirectoryEx(lpTemplateDirectory,
      lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12629:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
      bInitialState:BOOL;lpName:PKOLChar):THandle;
12630:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
      PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12631:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
      dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12632:  Function wCreateHardLink(lpFileName,
      lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12633:  Function
      CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12634:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
      nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12635:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
      lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
      Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
      lpProcessInfo:TProcessInformation):BOOL
12636:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
      Longint; lpName : PKOLChar) : THandle
12637:  Function
      wCreateWaitableTimer(lpTimerAttribs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
```

```
12638:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12639:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12640:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12641:  //Function
        wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12642:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12643:  //Function
        wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12644:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12645:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12646:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12647:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12648:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12649:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12650:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
        dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12651:  Function wFindAtom( lpString : PKOLChar) : ATOM
12652:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12653:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12654:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12655:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12656:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12657:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12658:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12659:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12660:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12661:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12662:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12663:  Function wGetCommandLine : PKOLChar
12664:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12665:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12666:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12667:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12668:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12669:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12670:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12671:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12672:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12673:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12674:  Function wGetEnvironmentStrings : PKOLChar
12675:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12676:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12677:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12678:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12679:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12680:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12681:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12682:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12683:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12684:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12685:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12686:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12687:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12688:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
        nSize:DWORD; lpFileName : PKOLChar) : DWORD
12689:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12690:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12691:  Function wGetProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12692:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12693:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12694: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
        lpCharType):BOOL
12695:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12696:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12697:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12698: //Function
        wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12699: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12700: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
        : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
        lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12701:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12702:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12703:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12704:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
```

```
12705:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12706:  Function
        wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12707:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12708:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12709:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12710:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12711:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
        TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12712:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12713:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12714:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12715:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12716:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12717:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12718:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
        lpNumberOfEventsRead:DWORD):BOOL;
12719:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12720:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12721:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
        lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12722:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
        lpNumbOfEventsRead:DWORD):BOOL;
12723:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
        : TCoord; var lpReadRegion : TSmallRect) : BOOL
12724:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12725:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12726:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
        lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12727:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12728:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12729:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12730:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12731:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12732:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12733:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12734:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12735:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12736:  //Function wUpdateResource(hUpdate:THandle;lpType,
        lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12737:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12738:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12739:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
        DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12740:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
        var lpNumberOfEventsWritten : DWORD) : BOOL
12741:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
        TCoord; var lpWriteRegion : TSmallRect) : BOOL
12742:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12743:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12744:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12745:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12746:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12747:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12748:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12749:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12750:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12751:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12752:  Function wlstrlen( lpString : PKOLChar) : Integer
12753:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
        PNetConnectInfoStruct) : DWORD
12754:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12755:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12756:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12757:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12758:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12759:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12760:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12761:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12762:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12763:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
        : PKOLChar; nNameBufSize : DWORD) : DWORD
12764:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12765:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12766:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12767: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
12768:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12769: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12770: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12771:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12772:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
```

```
12773:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12774:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12775:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12776: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12777: //Function wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12778: //Function wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12779:  Function wAddFontResource( FileName : PKOLChar) : Integer
12780:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12781:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12782:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12783:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12784:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12785: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
12786:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
        fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
        fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12787:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12788:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12789: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12790:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12791:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12792: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
        pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12793: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12794:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12795:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12796:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12797:  //Function wExtTextOut(DC:HDC;X,
        Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12798:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12799:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12800:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12801:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12802: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12803: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12804:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12805:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12806: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12807: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
        lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12808:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12809: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12810:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12811: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12812:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12813:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12814:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12815:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12816:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12817:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12818:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12819:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12820:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12821:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12822:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12823:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12824:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12825:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12826:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12827:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12828:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12829:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12830:  //Function
        wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12831:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12832: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
        dwFlags : DWORD; lParam : Pointer) : Longint
12833:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12834:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12835:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12836:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12837:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12838:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12839: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12840:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12841:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12842:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12843:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12844:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12845:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12846:  //Function wCreateDesktop(lpszDesktop,
        lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12847:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
        HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12848:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
        lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
```

```
12849:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
        hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12850: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
        nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12851:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
        dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12852:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12853:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12854:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12855:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12856:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
        : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12857:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
        : TFNDlgProc; dwInitParam : LPARAM) : Integer
12858:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12859:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
        nIDStaticPath:Integer;uFileType:UINT):Integer;
12860:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
        nIDStaticPath:Int;uFiletype:UINT):Int;
12861:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12862:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12863: //Function wDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
        cy:Int;Flags:UINT):BOOL;
12864:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12865:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12866:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12867:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
        pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12868: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12869:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12870:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12871:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12872:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12873:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12874:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12875:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12876:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12877:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12878:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12879:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12880:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
        lpnTabStopPositions) : DWORD
12881: //Function wGetUserObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
        lpnLengthNeed:DWORD)BOOL;
12882:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12883:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12884:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12885:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12886: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
        nHeigt:Int):BOOL;
12887:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12888:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12889:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12890:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12891:  Function wIsCharLower( ch : KOLChar) : BOOL
12892:  Function wIsCharUpper( ch : KOLChar) : BOOL
12893:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12894:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12895:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12896:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12897:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12898:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12899:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12900:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12901:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12902:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12903:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12904:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12905:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
12906:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
12907:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12908:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
12909:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12910:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
12911:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
12912:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12913:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12914:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12915:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
12916:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
12917:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12918:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12919:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
12920: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
12921: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
12922:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
12923: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12924:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
12925:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
```

```
12926:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12927:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12928:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
        lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
12929:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
        lpdwResult:DWORD): LRESULT
12930:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12931:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
12932:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
12933:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12934:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
12935: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
12936:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
12937:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
12938:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
12939:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12940: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
12941:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
        lpnTabStopPositions,nTabOrigin:Int):Longint;
12942:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
12943:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
12944:  Function wVkKeyScan( ch : KOLChar) : SHORT
12945:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
12946:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
12947:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
12948:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
12949:
12950: //TestDrive!
12951: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
12952:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
12953:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
12954:  Function GetLocalUserSidStr( const UserName : string) : string
12955:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
12956:  Function Impersonate2User( const domain : string; const user : string) : boolean
12957:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
12958:  Function KillProcessbyname( const exename : string; var found : integer) : integer
12959:  Function getWinProcessList : TStringList
12960: end;
12961:
12962: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
12963: begin
12964:  'AfMaxSyncSlots','LongInt'( 64);
12965:  'AfSynchronizeTimeout','LongInt'( 2000);
12966:   TAfSyncSlotID', 'DWORD
12967:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
12968:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
12969:   TAfSafeDirectSyncEvent', 'Procedure
12970:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
12971:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
12972:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
12973:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
12974:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
12975:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
12976:  Function AfIsSyncMethod : Boolean
12977:  Function AfSyncWnd : HWnd
12978:  Function AfSyncStatistics : TAfSyncStatistics
12979:  Procedure AfClearSyncStatistics
12980: end;
12981:
12982: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
12983: begin
12984:  'fBinary','LongWord')( $00000001);
12985:  'fParity','LongWord'( $00000002);
12986:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
12987:  'fOutxDsrFlow','LongWord')( $00000008);
12988:  'fDtrControl','LongWord')( $00000030);
12989:  'fDtrControlDisable','LongWord')( $00000000);
12990:  'fDtrControlEnable','LongWord')( $00000010);
12991:  'fDtrControlHandshake','LongWord')( $00000020);
12992:  'fDsrSensitivity','LongWord')( $00000040);
12993:  'fTXContinueOnXoff','LongWord')( $00000080);
12994:  'fOutX','LongWord')( $00000100);
12995:  'fInX','LongWord')( $00000200);
12996:  'fErrorChar','LongWord')( $00000400);
12997:  'fNull','LongWord')( $00000800);
12998:  'fRtsControl','LongWord')( $00003000);
12999:  'fRtsControlDisable','LongWord')( $00000000);
13000:  'fRtsControlEnable','LongWord')( $00001000);
13001:  'fRtsControlHandshake','LongWord')( $00002000);
13002:  'fRtsControlToggle','LongWord')( $00003000);
13003:  'fAbortOnError','LongWord')( $00004000);
13004:  'fDummy2','LongWord')( $FFFF8000);
13005:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13006:   FindClass('TOBJECT'),'EAfComPortCoreError
13007:   FindClass('TOBJECT'),'TAfComPortCore
13008:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
13009:    +'tKind : TAfCoreEvent; Data : DWORD)
13010:   SIRegister_TAfComPortCoreThread(CL);
13011:   SIRegister_TAfComPortEventThread(CL);
```

```
13012:   SIRegister_TAfComPortWriteThread(CL);
13013:   SIRegister_TAfComPortCore(CL);
13014:  Function FormatDeviceName( PortNumber : Integer) : string
13015: end;
13016:
13017: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13018: begin
13019:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13020:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
13021:   SIRegister_TApplicationFileIO(CL);
13022:   TDataFileCapability', '( dfcRead, dfcWrite )
13023:   TDataFileCapabilities', 'set of TDataFileCapability
13024:   SIRegister_TDataFile(CL);
13025:   //TDataFileClass', 'class of TDataFile
13026:  Function ApplicationFileIODefined : Boolean
13027:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13028:  Function FileStreamExists(const fileName: String) : Boolean
13029:  //Procedure Register
13030: end;
13031:
13032: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13033: begin
13034:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13035:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13036:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13037:   TALFBXScale', 'Integer
13038:   FindClass('TOBJECT'),'EALFBXConvertError
13039:   SIRegister_EALFBXError(CL);
13040:   SIRegister_EALFBXException(CL);
13041:   FindClass('TOBJECT'),'EALFBXGFixError
13042:   FindClass('TOBJECT'),'EALFBXDSQLError
13043:   FindClass('TOBJECT'),'EALFBXDynError
13044:   FindClass('TOBJECT'),'EALFBXGBakError
13045:   FindClass('TOBJECT'),'EALFBXGSecError
13046:   FindClass('TOBJECT'),'EALFBXLicenseError
13047:   FindClass('TOBJECT'),'EALFBXGStatError
13048:   //EALFBXExceptionClass', 'class of EALFBXError
13049:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13050:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13051:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13052:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13053:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13054:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13055:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13056:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13057:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13058:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13059:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13060:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13061:   TALFBXTransParams', 'set of TALFBXTransParam
13062:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13063:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13064:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13065:  'cALFBXMaxParamLength','LongInt'( 125);
13066:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13067:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13068:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13069:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13070:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13071:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13072:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13073:   SIRegister_TALFBXSQLDA(CL);
13074:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13075:   SIRegister_TALFBXPoolStream(CL);
13076:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13077:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13078:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13079:   //TALFBXArrayDesc', 'TISCArrayDesc
13080:   //TALFBXBlobDesc', 'TISCBlobDesc
13081:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13082:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13083:   SIRegister_TALFBXSQLResult(CL);
13084:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13085:   SIRegister_TALFBXSQLParams(CL);
13086:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13087:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13088:    +'atementType : TALFBXStatementType; end
13089:   FindClass('TOBJECT'),'TALFBXLibrary
13090:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13091:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13092:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13093:    //+' Excep : EALFBXExceptionClass)
13094:   SIRegister_TALFBXLibrary(CL);
13095:  'cAlFBXDateOffset','LongInt'( 15018);
13096:  'cALFBXTimeCoeff','LongInt'( 864000000);
13097:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13098:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13099:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13100:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
```

```
13101:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13102:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13103:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13104:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13105:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13106:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13107:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13108:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13109:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13110:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13111: end;
13112:
13113: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13114: begin
13115:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13116:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13117:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13118:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13119:    +'teger; First : Integer; CacheThreshold : Integer; end
13120:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13121:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13122:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13123:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13124:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13125:   SIRegister_TALFBXClient(CL);
13126:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13127:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13128:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13129:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13130:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13131:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13132:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13133:   SIRegister_TALFBXConnectionPoolClient(CL);
13134:   SIRegister_TALFBXEventThread(CL);
13135:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13136: end;
13137:
13138: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13139: begin
13140: _OSVERSIONINFOA = record
13141:     dwOSVersionInfoSize: DWORD;
13142:     dwMajorVersion: DWORD;
13143:     dwMinorVersion: DWORD;
13144:     dwBuildNumber: DWORD;
13145:     dwPlatformId: DWORD;
13146:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13147:   end;
13148:  TOSVersionInfoA', '_OSVERSIONINFOA
13149:   TOSVersionInfo', 'TOSVersionInfoA
13150:  'WS_EX_RIGHT','LongWord')( $00001000);
13151:  'WS_EX_LEFT','LongWord')( $00000000);
13152:  'WS_EX_RTLREADING','LongWord')( $00002000);
13153:  'WS_EX_LTRREADING','LongWord')( $00000000);
13154:  'WS_EX_LEFTSCROLLBAR','LongWord')( $00004000);
13155:  'WS_EX_RIGHTSCROLLBAR','LongWord')( $00000000);
13156:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13157:  'LAYOUT_RTL','LongWord')( $00000001);
13158:  'LAYOUT_BTT','LongWord')( $00000002);
13159:  'LAYOUT_VBH','LongWord')( $00000004);
13160:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13161:  'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13162:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13163:  Function GetLayout( dc : hdc) : DWORD
13164:  Function IsBidi : Boolean
13165:  Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13166:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13167:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13168:  Function GetPriorityClass( hProcess : THandle) : DWORD
13169:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13170:  Function CloseClipboard : BOOL
13171:  Function GetClipboardSequenceNumber : DWORD
13172:  Function GetClipboardOwner : HWND
13173:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13174:  Function GetClipboardViewer : HWND
13175:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13176:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13177:  Function GetClipboardData( uFormat : UINT) : THandle
13178:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13179:  Function CountClipboardFormats : Integer
13180:  Function EnumClipboardFormats( format : UINT) : UINT
13181:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13182:  Function EmptyClipboard : BOOL
13183:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13184:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13185:  Function GetOpenClipboardWindow : HWND
13186:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13187:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13188:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13189:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
```

```
13190:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13191:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13192:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13193:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13194:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13195: end;
13196:
13197: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13198: begin
13199:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13200:  Function GetTemporaryFilesPath : String
13201:  Function GetTemporaryFileName : String
13202:  Function FindFileInPaths( const fileName, paths : String) : String
13203:  Function PathsToString( const paths : TStrings) : String
13204:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13205:  //Function MacroExpandPath( const aPath : String) : String
13206: end;
13207:
13208: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13209: begin
13210:   SIRegister_TALMultiPartBaseContent(CL);
13211:   SIRegister_TALMultiPartBaseContents(CL);
13212:   SIRegister_TAlMultiPartBaseStream(CL);
13213:   SIRegister_TALMultipartBaseEncoder(CL);
13214:   SIRegister_TALMultipartBaseDecoder(CL);
13215:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13216:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13217:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13218: end;
13219:
13220: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13221: begin
13222:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13223:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13224:    +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13225:  Function aAllocPadedMem( Size : Cardinal) : TObject
13226:  Procedure aFreePadedMem( var P : TObject);
13227:  Procedure aFreePadedMem1( var P : PChar);
13228:  Function aCheckPadedMem( P : Pointer) : Byte
13229:  Function aGetPadMemSize( P : Pointer) : Cardinal
13230:  Function aAllocMem( Size : Cardinal) : Pointer
13231:  Function aStrLen( const Str : PChar) : Cardinal
13232:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13233:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13234:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13235:  Function aStrEnd( const Str : PChar) : PChar
13236:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13237:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13238:  Function aPCharLength( const Str : PChar) : Cardinal
13239:  Function aPCharUpper( Str : PChar) : PChar
13240:  Function aPCharLower( Str : PChar) : PChar
13241:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13242:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13243:  Function aCopyTail( const S : String; Len : Integer) : String
13244:  Function aInt2Thos( I : Int64) : String
13245:  Function aUpperCase( const S : String) : String
13246:  Function aLowerCase( const S : string) : String
13247:  Function aCompareText( const S1, S2 : string) : Integer
13248:  Function aSameText( const S1, S2 : string) : Boolean
13249:  Function aInt2Str( Value : Int64) : String
13250:  Function aStr2Int( const Value : String) : Int64
13251:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13252:  Function aGetFileExt( const FileName : String) : String
13253:  Function aGetFilePath( const FileName : String) : String
13254:  Function aGetFileName( const FileName : String) : String
13255:  Function aChangeExt( const FileName, Extension : String) : String
13256:  Function aAdjustLineBreaks( const S : string) : string
13257:  Function aGetWindowStr( WinHandle : HWND) : String
13258:  Function aDiskSpace( Drive : String) : TdriveSize
13259:  Function aFileExists( FileName : String) : Boolean
13260:  Function aFileSize( FileName : String) : Int64
13261:  Function aDirectoryExists( const Name : string) : Boolean
13262:  Function aSysErrorMessage( ErrorCode : Integer) : string
13263:  Function aShortPathName( const LongName : string) : string
13264:  Function aGetWindowVer : TWinVerRec
13265:  procedure InitDriveSpacePtr;
13266: end;
13267:
13268: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13269: begin
13270:  aZero','LongInt'( 0);
13271:  'makeappDEF','LongInt'( - 1);
13272:   'CS_VREDRAW','LongInt'( DWORD ( 1 ));
13273:  'CS_HREDRAW','LongInt'( DWORD ( 2 ));
13274:  'CS_KEYCVTWINDOW','LongInt'( 4);
13275:  'CS_DBLCLKS','LongInt'( 8);
13276:  'CS_OWNDC','LongWord')( $20);
13277:  'CS_CLASSDC','LongWord')( $40);
13278:  'CS_PARENTDC','LongWord')( $80);
```

```
13279:  'CS_NOKEYCVT','LongWord')( $100);
13280:  'CS_NOCLOSE','LongWord')( $200);
13281:  'CS_SAVEBITS','LongWord')( $800);
13282:  'CS_BYTEALIGNCLIENT','LongWord')( $1000);
13283:  'CS_BYTEALIGNWINDOW','LongWord')( $2000);
13284:  'CS_GLOBALCLASS','LongWord')( $4000);
13285:  'CS_IME','LongWord')( $10000);
13286:  'CS_DROPSHADOW','LongWord')( $20000);
13287:   //PPanelFunc', '^TPanelFunc // will not work
13288:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13289:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13290:   TFontLooks', 'set of TFontLook
13291:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13292:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13293:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13294:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13295:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13296:  Procedure RunMsgLoop( Show : Boolean)
13297:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13298:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
       ID_Number:Cardinal;hFont:Int):Int;
13299:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13300:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13301:  Function MakePanel(Left,Top,Width,Height,
       hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13302:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13303:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13304:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13305: end;
13306:
13307: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13308: begin
13309:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13310:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13311:   TScreenSaverOptions', 'set of TScreenSaverOption
13312:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
       ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13313:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13314:   SIRegister_TScreenSaver(CL);
13315:  //Procedure Register
13316:  Procedure SetScreenSaverPassword
13317: end;
13318:
13319: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13320: begin
13321:   FindClass('TOBJECT'),'TXCollection
13322:   SIRegister_EFilerException(CL);
13323:   SIRegister_TXCollectionItem(CL);
13324:   //TXCollectionItemClass', 'class of TXCollectionItem
13325:   SIRegister_TXCollection(CL);
13326:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13327:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13328:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13329:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13330:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13331:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13332: end;
13333:
13334: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13335: begin
13336:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13337:  Procedure xglMapTexCoordToNull
13338:  Procedure xglMapTexCoordToMain
13339:  Procedure xglMapTexCoordToSecond
13340:  Procedure xglMapTexCoordToDual
13341:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13342:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13343:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13344:  Procedure xglBeginUpdate
13345:  Procedure xglEndUpdate
13346:  Procedure xglPushState
13347:  Procedure xglPopState
13348:  Procedure xglForbidSecondTextureUnit
13349:  Procedure xglAllowSecondTextureUnit
13350:  Function xglGetBitWiseMapping : Cardinal
13351: end;
13352:
13353: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13354: begin
13355:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13356:   TBaseListOptions', 'set of TBaseListOption
13357:   SIRegister_TBaseList(CL);
13358:   SIRegister_TBaseVectorList(CL);
13359:   SIRegister_TAffineVectorList(CL);
13360:   SIRegister_TVectorList(CL);
13361:   SIRegister_TTexPointList(CL);
13362:   SIRegister_TXIntegerList(CL);
13363:   //PSingleArrayList', '^TSingleArrayList // will not work
13364:   SIRegister_TSingleList(CL);
```

```
13365:   SIRegister_TByteList(CL);
13366:   SIRegister_TQuaternionList(CL);
13367:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13368:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13369:  Procedure FastQuickSortLists(startIndex,
       endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13370: end;
13371:
13372: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13373: begin
13374:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13375:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13376:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
       indices:TIntegerList;list:TAffineVectorList);
13377:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
       indices:TIntegerList;list:TAffineVectorList);
13378:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
       normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13379:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13380:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13381:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13382:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13383:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13384:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13385:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13386:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13387:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13388:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
       edgesTriangles : TIntegerList) : TIntegerList
13389:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13390:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
       TPersistentObjectList;
13391:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13392:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
       TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13393: end;
13394:
13395: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13396: begin
13397:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13398:  Procedure FreeMemAndNil( var P : TObject)
13399:  Function PCharOrNil( const S : string) : PChar
13400:   SIRegister_TJclReferenceMemoryStream(CL);
13401:   FindClass('TOBJECT'),'EJclVMTError
13402:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13403:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13404:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13405:   PDynamicIndexList', '^TDynamicIndexList // will not work
13406:   PDynamicAddressList', '^TDynamicAddressList // will not work
13407:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13408:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13409:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13410:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13411:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13412:  Function GetInitTable( AClass : TClass) : PTypeInfo
13413:   PFieldEntry', '^TFieldEntry // will not work}
13414:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : ShortString; end
13415:  Function JIsClass( Address : Pointer) : Boolean
13416:  Function JIsObject( Address : Pointer) : Boolean
13417:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13418:   TDigitCount', 'Integer
13419:   SIRegister_TJclNumericFormat(CL);
13420:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13421:   TTextHandler', 'Procedure ( const Text : string)
13422: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223);
13423:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13424:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13425:  Function ReadKey : Char   //to and from the DOS console !
13426:   TModuleHandle', 'HINST
13427:   //TModuleHandle', 'Pointer
13428:  'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ));
13429:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13430:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13431:  Procedure UnloadModule( var Module : TModuleHandle)
13432:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13433:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13434:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13435:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13436:   FindClass('TOBJECT'),'EJclConversionError
13437:  Function JStrToBoolean( const S : string) : Boolean
13438:  Function JBooleanToStr( B : Boolean) : string
13439:  Function JIntToBool( I : Integer) : Boolean
13440:  Function JBoolToInt( B : Boolean) : Integer
13441:  'ListSeparator','String ';
13442:  'ListSeparator1','String ':
13443:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13444:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13445:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
```

```
13446:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13447:  Function ListItemCount( const List, Separator : string) : Integer
13448:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13449:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13450:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13451:  Function SystemTObjectInstance : LongWord
13452:  Function IsCompiledWithPackages : Boolean
13453:  Function JJclGUIDToString( const GUID : TGUID) : string
13454:  Function JJclStringToGUID( const S : string) : TGUID
13455:   SIRegister_TJclIntfCriticalSection(CL);
13456:   SIRegister_TJclSimpleLog(CL);
13457:  Procedure InitSimpleLog( const ALogFileName : string)
13458: end;
13459:
13460: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13461: begin
13462:   FindClass('TOBJECT'),'EJclBorRADException
13463:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13464:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13465:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13466:   TJclBorRADToolPath', 'string
13467:  'SupportedDelphiVersions','LongInt'( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13468:  'SupportedBCBVersions','LongInt'( 5 or  6 or  10 or  11);
13469:  'SupportedBDSVersions','LongInt'( 1 or  2 or  3 or  4 or  5);
13470:  BorRADToolRepositoryPagesSection','String 'Repository Pages
13471:  BorRADToolRepositoryDialogsPage','String 'Dialogs
13472:  BorRADToolRepositoryFormsPage','String 'Forms
13473:  BorRADToolRepositoryProjectsPage','String 'Projects
13474:  BorRADToolRepositoryDataModulesPage','String 'Data Modules
13475:  BorRADToolRepositoryObjectType','String 'Type
13476:  BorRADToolRepositoryFormTemplate','String 'FormTemplate
13477:  BorRADToolRepositoryProjectTemplate','String 'ProjectTemplate
13478:  BorRADToolRepositoryObjectName','String 'Name
13479:  BorRADToolRepositoryObjectPage','String 'Page
13480:  BorRADToolRepositoryObjectIcon','String 'Icon
13481:  BorRADToolRepositoryObjectDescr','String 'Description
13482:  BorRADToolRepositoryObjectAuthor','String 'Author
13483:  BorRADToolRepositoryObjectAncestor','String 'Ancestor
13484:  BorRADToolRepositoryObjectDesigner','String 'Designer
13485:  BorRADToolRepositoryDesignerDfm','String 'dfm
13486:  BorRADToolRepositoryDesignerXfm','String 'xfm
13487:  BorRADToolRepositoryObjectNewForm','String 'DefaultNewForm
13488:  BorRADToolRepositoryObjectMainForm','String 'DefaultMainForm
13489:  SourceExtensionDelphiPackage','String '.dpk
13490:  SourceExtensionBCBPackage','String '.bpk
13491:  SourceExtensionDelphiProject','String '.dpr
13492:  SourceExtensionBCBProject','String '.bpr
13493:  SourceExtensionBDSProject','String '.bdsproj
13494:  SourceExtensionDProject','String '.dproj
13495:  BinaryExtensionPackage','String '.bpl
13496:  BinaryExtensionLibrary','String '.dll
13497:  BinaryExtensionExecutable','String '.exe
13498:  CompilerExtensionDCP','String '.dcp
13499:  CompilerExtensionBPI','String '.bpi
13500:  CompilerExtensionLIB','String '.lib
13501:  CompilerExtensionTDS','String '.tds
13502:  CompilerExtensionMAP','String '.map
13503:  CompilerExtensionDRC','String '.drc
13504:  CompilerExtensionDEF','String '.def
13505:  SourceExtensionCPP','String '.cpp
13506:  SourceExtensionH','String '.h
13507:  SourceExtensionPAS','String '.pas
13508:  SourceExtensionDFM','String '.dfm
13509:  SourceExtensionXFM','String '.xfm
13510:  SourceDescriptionPAS','String 'Pascal source file
13511:  SourceDescriptionCPP','String 'C++ source file
13512:  DesignerVCL','String 'VCL
13513:  DesignerCLX','String 'CLX
13514:  ProjectTypePackage','String 'package
13515:  ProjectTypeLibrary','String 'library
13516:  ProjectTypeProgram','String 'program
13517:  Personality32Bit','String '32 bit
13518:  Personality64Bit','String '64 bit
13519:  PersonalityDelphi','String 'Delphi
13520:  PersonalityDelphiDotNet','String 'Delphi.net
13521:  PersonalityBCB','String 'C++Builder
13522:  PersonalityCSB','String 'C#Builder
13523:  PersonalityVB','String 'Visual Basic
13524:  PersonalityDesign','String 'Design
13525:  PersonalityUnknown','String 'Unknown personality
13526:  PersonalityBDS','String 'Borland Developer Studio
13527:  DOFDirectoriesSection','String 'Directories
13528:  DOFUnitOutputDirKey','String 'UnitOutputDir
13529:  DOFSearchPathName','String 'SearchPath
13530:  DOFConditionals','String 'Conditionals
13531:  DOFLinkerSection','String 'Linker
13532:  DOFPackagesKey','String 'Packages
13533:  DOFCompilerSection','String 'Compiler
13534:  DOFPackageNoLinkKey','String 'PackageNoLink
```

```
13535:  DOFAdditionalSection','String 'Additional
13536:  DOFOptionsKey','String 'Options
13537:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13538:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13539:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13540:   TJclBorPersonalities', 'set of TJclBorPersonality
13541:   TJclBorDesigner', '( bdVCL, bdCLX )
13542:   TJclBorDesigners', 'set of TJClBorDesigner
13543:   TJclBorPlatform', '( bp32bit, bp64bit )
13544:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13545:   SIRegister_TJclBorRADToolInstallationObject(CL);
13546:   SIRegister_TJclBorlandOpenHelp(CL);
13547:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13548:   TJclHelp2Objects', 'set of TJclHelp2Object
13549:   SIRegister_TJclHelp2Manager(CL);
13550:   SIRegister_TJclBorRADToolIdeTool(CL);
13551:   SIRegister_TJclBorRADToolIdePackages(CL);
13552:   SIRegister_IJclCommandLineTool(CL);
13553:   FindClass('TOBJECT'),'EJclCommandLineToolError
13554:   SIRegister_TJclCommandLineTool(CL);
13555:   SIRegister_TJclBorlandCommandLineTool(CL);
13556:   SIRegister_TJclBCC32(CL);
13557:   SIRegister_TJclDCC32(CL);
13558:   TJclDCC', 'TJclDCC32
13559:   SIRegister_TJclBpr2Mak(CL);
13560:   SIRegister_TJclBorlandMake(CL);
13561:   SIRegister_TJclBorRADToolPalette(CL);
13562:   SIRegister_TJclBorRADToolRepository(CL);
13563:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13564:   TCommandLineTools', 'set of TCommandLineTool
13565:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13566:   SIRegister_TJclBorRADToolInstallation(CL);
13567:   SIRegister_TJclBCBInstallation(CL);
13568:   SIRegister_TJclDelphiInstallation(CL);
13569:   SIRegister_TJclDCCIL(CL);
13570:   SIRegister_TJclBDSInstallation(CL);
13571:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13572:   SIRegister_TJclBorRADToolInstallations(CL);
13573:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13574:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13575:  Function IsDelphiPackage( const FileName : string) : Boolean
13576:  Function IsDelphiProject( const FileName : string) : Boolean
13577:  Function IsBCBPackage( const FileName : string) : Boolean
13578:  Function IsBCBProject( const FileName : string) : Boolean
13579:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13580:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13581:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
      Descript:PString;
13582:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
      Descript:PString
13583:  function SamePath(const Path1, Path2: string): Boolean;
13584: end;
13585:
13586: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13587: begin
13588:  'ERROR_NO_MORE_FILES','LongInt'( 18);
13589:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13590:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13591:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13592:  'LPathSeparator','String '/
13593:  'LDirDelimiter','String '/
13594:  'LDirSeparator','String ':
13595:  'JXPathDevicePrefix','String '\\.\
13596:  'JXPathSeparator','String '\
13597:  'JXDirDelimiter','String '\
13598:  'JXDirSeparator','String ';
13599:  'JXPathUncPrefix','String '\\
13600:  'faNormalFile','LongWord')( $00000080);
13601:  //'faUnixSpecific',' faSymLink);
13602:   JXTCompactPath', '( cpCenter, cpEnd )
13603:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13604:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13605:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13606:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13607:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13608:
13609:  Function jxPathAddSeparator( const Path : string) : string
13610:  Function jxPathAddExtension( const Path, Extension : string) : string
13611:  Function jxPathAppend( const Path, Append : string) : string
13612:  Function jxPathBuildRoot( const Drive : Byte) : string
13613:  Function jxPathCanonicalize( const Path : string) : string
13614:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13615:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13616:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13617:  Function jxPathExtractFileDirFixed( const S : string) : string
13618:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13619:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13620:  Function jxPathGetDepth( const Path : string) : Integer
13621:  Function jxPathGetLongName( const Path : string) : string
```

```
13622:  Function jxPathGetShortName( const Path : string) : string
13623:  Function jxPathGetLongName( const Path : string) : string
13624:  Function jxPathGetShortName( const Path : string) : string
13625:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13626:  Function jxPathGetTempPath : string
13627:  Function jxPathIsAbsolute( const Path : string) : Boolean
13628:  Function jxPathIsChild( const Path, Base : string) : Boolean
13629:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13630:  Function jxPathIsUNC( const Path : string) : Boolean
13631:  Function jxPathRemoveSeparator( const Path : string) : string
13632:  Function jxPathRemoveExtension( const Path : string) : string
13633:  Function jxPathGetPhysicalPath ( const LocalizedPath : string) : string
13634:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13635:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13636:   JxTFileListOptions', 'set of TFileListOption
13637:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13638:   TFileHandler', 'Procedure ( const FileName : string)
13639:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13640:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13641:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
       AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
       FileMatchFunc:TFileMatchFunc):Bool;
13642:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13643:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13644:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13645:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13646:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
       RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13647:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
       IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13648:  Procedure jxCreateEmptyFile( const FileName : string)
13649:  Function jxCloseVolume( var Volume : THandle) : Boolean
13650:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13651:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13652:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13653:  Function jxDelTree( const Path : string) : Boolean
13654:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13655:  Function jxDiskInDrive( Drive : Char) : Boolean
13656:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13657:  Function jxFileCreateTemp( var Prefix : string) : THandle
13658:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13659:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13660:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13661:  Function jxFileExists( const FileName : string) : Boolean
13662:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13663:  Function jxFileRestore( const FileName : string) : Boolean
13664:  Function jxGetBackupFileName( const FileName : string) : string
13665:  Function jxIsBackupFileName( const FileName : string) : Boolean
13666:  Function jxFileGetDisplayName( const FileName : string) : string
13667:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13668:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13669:  Function jxFileGetSize( const FileName : string) : Int64
13670:  Function jxFileGetTempName( const Prefix : string) : string
13671:  Function jxFileGetTypeName( const FileName : string) : string
13672:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13673:  Function jxForceDirectories( Name : string) : Boolean
13674:  Function jxGetDirectorySize( const Path : string) : Int64
13675:  Function jxGetDriveTypeStr( const Drive : Char) : string
13676:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13677:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13678:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13679:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13680:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13681:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
       ResolveSymLinks:Boolean):Integer
13682:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13683:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13684:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13685:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13686:  Function jxGetFileCreation( const FName : string) : TFileTime;
13687:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13688:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13689:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13690:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13691:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13692:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13693:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13694:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13695:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13696:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13697:  Function jxGetModulePath( const Module : HMODULE) : string
13698:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13699:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13700:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13701:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13702:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13703:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13704:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13705:  Function jxOpenVolume( const Drive : Char) : THandle
```

```
13706:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13707:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13708:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13709:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13710:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13711:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13712:  Procedure jxShredFile( const FileName : string; Times : Integer)
13713:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13714:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13715:  Function jxSymbolicLinkTarget( const Name : string) : string
13716:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13717:   SIRegister_TJclCustomFileAttrMask(CL);
13718:   SIRegister_TJclFileAttributeMask(CL);
13719:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13720:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13721:   TFileSearchOptions', 'set of TFileSearchOption
13722:   TFileSearchTaskID', 'Integer
13723:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13724:    +'hTaskID; const Aborted : Boolean)
13725:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13726:   SIRegister_IJclFileEnumerator(CL);
13727:   SIRegister_TJclFileEnumerator(CL);
13728:  Function JxFileSearch : IJclFileEnumerator
13729:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13730:   JxTFileFlags', 'set of TFileFlag
13731:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13732:   SIRegister_TJclFileVersionInfo(CL);
13733:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13734:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13735:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13736:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13737:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13738:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13739:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13740:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13741:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13742:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13743:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13744:   SIRegister_TJclTempFileStream(CL);
13745:   FindClass('TOBJECT'),'TJclCustomFileMapping
13746:   SIRegister_TJclFileMappingView(CL);
13747:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13748:   SIRegister_TJclCustomFileMapping(CL);
13749:   SIRegister_TJclFileMapping(CL);
13750:   SIRegister_TJclSwapFileMapping(CL);
13751:   SIRegister_TJclFileMappingStream(CL);
13752:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13753:   //PPCharArray', '^TPCharArray // will not work
13754:   SIRegister_TJclMappedTextReader(CL);
13755:   SIRegister_TJclFileMaskComparator(CL);
13756:   FindClass('TOBJECT'),'EJclPathError
13757:   FindClass('TOBJECT'),'EJclFileUtilsError
13758:   FindClass('TOBJECT'),'EJclTempFileStreamError
13759:   FindClass('TOBJECT'),'EJclTempFileStreamError
13760:   FindClass('TOBJECT'),'EJclFileMappingError
13761:   FindClass('TOBJECT'),'EJclFileMappingViewError
13762:  Function jxPathGetLongName2( const Path : string) : string
13763:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13764:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13765:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13766:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13767:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13768:  Procedure jxPathListAddItems( var List : string; const Items : string)
13769:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13770:  Procedure jxPathListDelItems( var List : string; const Items : string)
13771:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13772:  Function jxPathListItemCount( const List : string) : Integer
13773:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13774:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13775:  Function jxPathListItemIndex( const List, Item : string) : Integer
13776:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13777:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13778:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13779:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string) : Integer
13780: end;
13781:
13782: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13783: begin
13784:  'UTF8FileHeader','String #$ef#$bb#$bf);
13785:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13786:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13787:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13788:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13789:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13790:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
```

```
13791:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13792:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13793:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13794:  Function lFileIsReadable( const AFilename : string) : boolean
13795:  Function lFileIsWritable( const AFilename : string) : boolean
13796:  Function lFileIsText( const AFilename : string) : boolean
13797:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13798:  Function lFileIsExecutable( const AFilename : string) : boolean
13799:  Function lFileIsSymlink( const AFilename : string) : boolean
13800:  Function lFileIsHardLink( const AFilename : string) : boolean
13801:  Function lFileSize( const Filename : string) : int64;
13802:  Function lGetFileDescription( const AFilename : string) : string
13803:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13804:  Function lTryReadAllLinks( const Filename : string) : string
13805:  Function lDirPathExists( const FileName : String) : Boolean
13806:  Function lForceDirectory( DirectoryName : string) : boolean
13807:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13808:  Function lProgramDirectory : string
13809:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13810:  Function lExtractFileNameOnly( const AFilename : string) : string
13811:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13812:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13813:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13814:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13815:  Function lAppendPathDelim( const Path : string) : string
13816:  Function lChompPathDelim( const Path : string) : string
13817:  Function lTrimFilename( const AFilename : string) : string
13818:  Function lCleanAndExpandFilename( const Filename : string) : string
13819:  Function lCleanAndExpandDirectory( const Filename : string) : string
13820:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13821:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
       AlwaysRequireSharedBaseFolder : Boolean) : string
13822:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13823:  Function lFileIsInPath( const Filename, Path : string) : boolean
13824:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13825:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13826:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13827:  'AllDirectoryEntriesMask','String '*
13828:  Function lGetAllFilesMask : string
13829:  Function lGetExeExt : string
13830:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
       TSearchFileInPathFlags) : string
13831:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
       TSearchFileInPathFlags) : TStrings
13832:  Function lFindDiskFilename( const Filename : string) : string
13833:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13834:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13835:  Function lGetDarwinSystemFilename( Filename : string) : string
13836:   SIRegister_TFileIterator(CL);
13837:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13838:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13839:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13840:   SIRegister_TFileSearcher(CL);
13841:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13842:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13843:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13844:  // TCopyFileFlags', 'set of TCopyFileFlag
13845:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13846:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13847:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13848:  Function lReadFileToString( const Filename : string) : string
13849:  Function lGetTempFilename( const Directory, Prefix : string) : string
13850:  {Function NeedRTLAnsi : boolean
13851:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13852:  Function UTF8ToSys( const s : string) : string
13853:  Function SysToUTF8( const s : string) : string
13854:  Function ConsoleToUTF8( const s : string) : string
13855:  Function UTF8ToConsole( const s : string) : string}
13856:  Function FileExistsUTF8( const Filename : string) : boolean
13857:  Function FileAgeUTF8( const FileName : string) : Longint
13858:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13859:  Function ExpandFileNameUTF8( const FileName : string) : string
13860:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13861:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13862:  Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13863:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13864:  Procedure FindCloseUTF8( var F : TSearchrec)
13865:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13866:  Function FileGetAttrUTF8( const FileName : String) : Longint
13867:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13868:  Function DeleteFileUTF8( const FileName : String) : Boolean
13869:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13870:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13871:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13872:  Function GetCurrentDirUTF8 : String
13873:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13874:  Function CreateDirUTF8( const NewDir : String) : Boolean
13875:  Function RemoveDirUTF8( const Dir : String) : Boolean
13876:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
```

```
13877:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13878:  Function FileCreateUTF8( const FileName : string) : THandle;
13879:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13880:  Function ParamStrUTF8( Param : Integer) : string
13881:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13882:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13883:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13884:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13885:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13886: end;
13887:
13888: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13889: begin
13890:   //VK_F23 = 134;
13891:   //{$EXTERNALSYM VK_F24}
13892:   //VK_F24 = 135;
13893:  TVirtualKeyCode', 'Integer
13894:  'VK_MOUSEWHEELUP','integer'(134);
13895:  'VK_MOUSEWHEELDOWN','integer'(135);
13896:  Function glIsKeyDown( c : Char) : Boolean;
13897:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13898:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13899:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13900:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13901:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13902:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13903: end;
13904:
13905: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13906: begin
13907:   TGLPoint', 'TPoint
13908:   //PGLPoint', '^TGLPoint // will not work
13909:   TGLRect', 'TRect
13910:   //PGLRect', '^TGLRect // will not work
13911:   TDelphiColor', 'TColor
13912:   TGLPicture', 'TPicture
13913:   TGLGraphic', 'TGraphic
13914:   TGLBitmap', 'TBitmap
13915:   //TGraphicClass', 'class of TGraphic
13916:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13917:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
13918:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
13919:    +'Button; Shift : TShiftState; X, Y : Integer)
13920:   TGLMouseMoveEvent', 'TMouseMoveEvent
13921:   TGLKeyEvent', 'TKeyEvent
13922:   TGLKeyPressEvent', 'TKeyPressEvent
13923:   EGLOSError', 'EWin32Error
13924:   EGLOSError', 'EWin32Error
13925:   EGLOSError', 'EOSError
13926:  'glsAllFilter','string'All // sAllFilter
13927:  Function GLPoint( const x, y : Integer) : TGLPoint
13928:  Function GLRGB( const r, g, b : Byte) : TColor
13929:  Function GLColorToRGB( color : TColor) : TColor
13930:  Function GLGetRValue( rgb : DWORD) : Byte
13931:  Function GLGetGValue( rgb : DWORD) : Byte
13932:  Function GLGetBValue( rgb : DWORD) : Byte
13933:  Procedure GLInitWinColors
13934:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13935:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13936:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13937:  Procedure GLInformationDlg( const msg : String)
13938:  Function GLQuestionDlg( const msg : String) : Boolean
13939:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
13940:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
13941:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
13942:  Function GLApplicationTerminated : Boolean
13943:  Procedure GLRaiseLastOSError
13944:  Procedure GLFreeAndNil( var anObject: TObject)
13945:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
13946:  Function GLGetCurrentColorDepth : Integer
13947:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
13948:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
13949:  Procedure GLSleep( length : Cardinal)
13950:  Procedure GLQueryPerformanceCounter( var val : Int64)
13951:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
13952:  Function GLStartPrecisionTimer : Int64
13953:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
13954:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
13955:  Function GLRDTSC : Int64
13956:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
13957:  Function GLOKMessageBox( const Text, Caption : string) : Integer
13958:  Procedure GLShowHTMLUrl( Url : String)
13959:  Procedure GLShowCursor( AShow : boolean)
13960:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
13961:  Procedure GLGetCursorPos( var point : TGLPoint)
13962:  Function GLGetScreenWidth : integer
13963:  Function GLGetScreenHeight : integer
13964:  Function GLGetTickCount : int64
13965:  function RemoveSpaces(const str : String) : String;
```

```
13966:   TNormalMapSpace','( nmsObject, nmsTangent )
13967:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
13968:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
13969:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
        TAffineVectorList; Colors : TVectorList)
13970:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
        HiTexCoords:TAffineVectorList):TGLBitmap
13971:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
        LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
13972: end;
13973:
13974: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
13975: begin
13976:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
13977:  // PGLStarRecord', '^TGLStarRecord // will not work
13978:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
13979:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
13980:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
13981: end;
13982:
13983:
13984: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
13985: begin
13986:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
13987:   //PAABB', '^TAABB // will not work
13988:  TBSphere', 'record Center : TAffineVector; Radius : single; end
13989:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
13990:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
13991:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
13992:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
13993:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
13994:  Procedure SetAABB( var bb : TAABB; const v : TVector)
13995:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
13996:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
13997:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
13998:  Function BBMinX( const c : THmgBoundingBox) : Single
13999:  Function BBMaxX( const c : THmgBoundingBox) : Single
14000:  Function BBMinY( const c : THmgBoundingBox) : Single
14001:  Function BBMaxY( const c : THmgBoundingBox) : Single
14002:  Function BBMinZ( const c : THmgBoundingBox) : Single
14003:  Function BBMaxZ( const c : THmgBoundingBox) : Single
14004:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
14005:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14006:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
14007:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
14008:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
14009:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14010:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14011:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14012:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14013:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14014:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14015:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14016:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14017:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14018:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14019:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14020:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14021:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14022:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14023:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14024:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14025:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14026:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14027:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14028:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14029:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14030:  Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
        testBSphere:TBSphere):TSpaceContains
14031:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14032:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14033:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14034:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14035:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14036:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
        viewportSizeY:Int):TClipRect
14037: end;
14038:
14039: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14040: begin
14041:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14042:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14043:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14044:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14045:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14046:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14047:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14048:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14049:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
```

```
14050:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14051:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14052:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14053:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14054:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14055:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14056:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14057:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14058:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14059:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14060:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14061:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14062:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14063:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14064:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14065:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
14066: end;
14067:
14068: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14069: begin
14070:  'EPSILON','Single').setExtended( 1e-40);
14071:  'EPSILON2','Single').setExtended( 1e-30);  }
14072:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14073:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14074:  THmgPlane', 'TVector
14075:   TDoubleHmgPlane', 'THomogeneousDblVector
14076:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14077:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14078:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14079:   TSingleArray', 'array of Single
14080:   TTransformations','array [0..15] of Single)
14081:   TPackedRotationMatrix','array [0..2] of Smallint)
14082:   TVertex', 'TAffineVector
14083:   //TVectorGL', 'THomogeneousFltVector
14084:   //TMatrixGL', 'THomogeneousFltMatrix
14085:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14086:  Function glTexPointMake( const s, t : Single) : TTexPoint
14087:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14088:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14089:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14090:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14091:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14092:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14093:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14094:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14095:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14096:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14097:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14098:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14099:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14100:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14101:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14102:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14103:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14104:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14105:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14106:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14107:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14108:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14109:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14110:  Procedure glRstVector( var v : TAffineVector);
14111:  Procedure glRstVector1( var v : TVectorGL);
14112:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14113:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14114:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14115:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14116:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14117:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14118:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14119:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14120:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14121:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14122:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14123:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14124:  //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Int;dest:PTexPointArray);
14125:  //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
      nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14126:  //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
      PAffineVectorArray);
14127:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14128:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14129:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14130:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14131:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14132:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14133:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14134:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14135:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
```

```
14136:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14137:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14138:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14139:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14140:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14141:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14142:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14143:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14144:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14145:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14146:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14147:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14148:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14149:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14150:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14151:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14152:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14153:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14154:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14155:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14156:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14157:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14158:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14159:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14160:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14161:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14162:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14163:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14164:  Function glLerp( const start, stop, t : Single) : Single
14165:  Function glAngleLerp( start, stop, t : Single) : Single
14166:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14167:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14168:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14169:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14170:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14171:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14172:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14173:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14174: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14175: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
       PAffineVectorArray);
14176:  Function glVectorLength( const x, y : Single) : Single;
14177:  Function glVectorLength1( const x, y, z : Single) : Single;
14178:  Function glVectorLength2( const v : TAffineVector) : Single;
14179:  Function glVectorLength3( const v : TVectorGL) : Single;
14180:  Function glVectorLength4( const v : array of Single) : Single;
14181:  Function glVectorNorm( const x, y : Single) : Single;
14182:  Function glVectorNorm1( const v : TAffineVector) : Single;
14183:  Function glVectorNorm2( const v : TVectorGL) : Single;
14184:  Function glVectorNorm3( var V : array of Single) : Single;
14185:  Procedure glNormalizeVector( var v : TAffineVector);
14186:  Procedure glNormalizeVector1( var v : TVectorGL);
14187:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14188:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14189: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14190:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14191:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14192:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14193:  Procedure glNegateVector( var V : TAffineVector);
14194:  Procedure glNegateVector2( var V : TVectorGL);
14195:  Procedure glNegateVector3( var V : array of Single);
14196:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14197:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14198:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14199:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14200:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14201:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14202:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14203:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14204:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14205:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14206:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14207:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14208:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14209:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14210:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14211:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14212:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14213:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14214:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14215:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14216:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14217:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14218:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14219:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14220:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14221:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14222:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14223:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
```

```
14224:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14225:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14226:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14227:  Procedure glAbsVector( var v : TVectorGL);
14228:  Procedure glAbsVector1( var v : TAffineVector);
14229:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14230:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14231:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14232:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14233:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14234:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14235:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14236:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14237:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14238:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14239:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14240:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14241:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14242:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14243:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14244:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14245:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14246:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14247:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14248:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14249:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14250:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14251:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14252:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14253:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14254:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14255:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14256:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14257:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14258:  Procedure glAdjointMatrix( var M : TMatrixGL);
14259:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14260:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14261:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14262:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14263:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14264:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14265:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14266:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14267:  Procedure glInvertMatrix( var M : TMatrixGL);
14268:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14269:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14270:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14271:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14272:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14273:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14274:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14275:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14276:  Procedure glNormalizePlane( var plane : THmgPlane)
14277:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14278:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14279:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14280:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14281:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14282:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14283:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14284:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14285:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14286:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14287:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14288:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14289:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14290:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
       Segment0Closest, Segment1Closest : TAffineVector)
14291:  Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14292:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14293:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14294:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14295:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14296:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14297:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14298:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14299:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14300:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14301:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14302:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14303:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14304:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14305:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14306:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14307:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14308:  Function glLnXP1( X : Extended) : Extended
14309:  Function glLog10( X : Extended) : Extended
14310:  Function glLog2( X : Extended) : Extended;
14311:  Function glLog21( X : Single) : Single;
```

```
14312:  Function glLogN( Base, X : Extended) : Extended
14313:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14314:  Function glPower( const Base, Exponent : Single) : Single;
14315:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14316:  Function glDegToRad( const Degrees : Extended) : Extended;
14317:  Function glDegToRad1( const Degrees : Single) : Single;
14318:  Function glRadToDeg( const Radians : Extended) : Extended;
14319:  Function glRadToDeg1( const Radians : Single) : Single;
14320:  Function glNormalizeAngle( angle : Single) : Single
14321:  Function glNormalizeDegAngle( angle : Single) : Single
14322:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14323:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14324:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14325:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14326:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14327:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14328:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14329:  Function glArcCos( const X : Extended) : Extended;
14330:  Function glArcCos1( const x : Single) : Single;
14331:  Function glArcSin( const X : Extended) : Extended;
14332:  Function glArcSin1( const X : Single) : Single;
14333:  Function glArcTan21( const Y, X : Extended) : Extended;
14334:  Function glArcTan21( const Y, X : Single) : Single;
14335:  Function glFastArcTan2( y, x : Single) : Single
14336:  Function glTan( const X : Extended) : Extended;
14337:  Function glTan1( const X : Single) : Single;
14338:  Function glCoTan( const X : Extended) : Extended;
14339:  Function glCoTan1( const X : Single) : Single;
14340:  Function glSinh( const x : Single) : Single;
14341:  Function glSinh1( const x : Double) : Double;
14342:  Function glCosh( const x : Single) : Single;
14343:  Function glCosh1( const x : Double) : Double;
14344:  Function glRSqrt( v : Single) : Single
14345:  Function glRLength( x, y : Single) : Single
14346:  Function glISqrt( i : Integer) : Integer
14347:  Function glILength( x, y : Integer) : Integer;
14348:  Function glILength1( x, y, z : Integer) : Integer;
14349:  Procedure glRegisterBasedExp
14350:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14351:  Function glRoundInt( v : Single) : Single;
14352:  Function glRoundInt1( v : Extended) : Extended;
14353:  Function glTrunc( v : Single) : Integer;
14354:  Function glTrunc64( v : Extended) : Int64;
14355:  Function glInt( v : Single) : Single;
14356:  Function glInt1( v : Extended) : Extended;
14357:  Function glFrac( v : Single) : Single;
14358:  Function glFrac1( v : Extended) : Extended;
14359:  Function glRound( v : Single) : Integer;
14360:  Function glRound64( v : Single) : Int64;
14361:  Function glRound641( v : Extended) : Int64;
14362:  Function glTrunc( X : Extended) : Int64
14363:  Function glRound( X : Extended) : Int64
14364:  Function glFrac( X : Extended) : Extended
14365:  Function glCeil( v : Single) : Integer;
14366:  Function glCeil64( v : Extended) : Int64;
14367:  Function glFloor( v : Single) : Integer;
14368:  Function glFloor64( v : Extended) : Int64;
14369:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14370:  Function glSign( x : Single) : Integer
14371:  Function glIsInRange( const x, a, b : Single) : Boolean;
14372:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14373:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14374:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14375:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14376:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14377:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14378:  Function glMinFloat3( const v1, v2 : Single) : Single;
14379:  Function glMinFloat4( const v : array of Single) : Single;
14380:  Function glMinFloat5( const v1, v2 : Double) : Double;
14381:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14382:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14383:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14384:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14385:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14386:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14387:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14388:  Function glMaxFloat2( const v : array of Single) : Single;
14389:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14390:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14391:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14392:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14393:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14394:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14395:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14396:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14397:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14398:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14399:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14400:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
```

```
14401:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14402:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14403:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14404:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14405:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14406:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14407:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14408:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14409:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14410:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14411:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14412:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14413:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14414:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14415:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14416:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14417:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14418:  Procedure glSortArrayAscending( var a : array of Extended)
14419:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14420:  Function glClampValue1( const aValue, aMin : Single) : Single;
14421:  Function glGeometryOptimizationMode : String
14422:  Procedure glBeginFPUOnlySection
14423:  Procedure glEndFPUOnlySection
14424:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14425:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14426:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14427:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14428:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14429:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14430:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14431:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14432:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14433:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14434:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14435:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14436:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14437:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14438:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14439:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14440:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14441:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14442:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14443:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14444:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14445:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14446:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14447:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14448:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14449:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14450:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14451:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14452:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14453:  'cPI','Single').setExtended( 3.141592654);
14454:  'cPIdiv180','Single').setExtended( 0.017453292);
14455:  'c180divPI','Single').setExtended( 57.29577951);
14456:  'c2PI','Single').setExtended( 6.283185307);
14457:  'cPIdiv2','Single').setExtended( 1.570796326);
14458:  'cPIdiv4','Single').setExtended( 0.785398163);
14459:  'c3PIdiv4','Single').setExtended( 2.35619449);
14460:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14461:  'cInv360','Single').setExtended( 1 / 360);
14462:  'c180','Single').setExtended( 180);
14463:  'c360','Single').setExtended( 360);
14464:  'cOneHalf','Single').setExtended( 0.5);
14465:  'cLn10','Single').setExtended( 2.302585093);
14466:  {'MinSingle','Extended').setExtended( 1.5e-45);
14467:  'MaxSingle','Extended').setExtended( 3.4e+38);
14468:  'MinDouble','Extended').setExtended( 5.0e-324);
14469:  'MaxDouble','Extended').setExtended( 1.7e+308);
14470:  'MinExtended','Extended').setExtended( 3.4e-4932);
14471:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14472:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14473:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14474: end;
14475:
14476: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14477: begin
14478:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14479:   (FindClass('TOBJECT'),'TFaceGroups
14480:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14481:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14482:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14483:   SIRegister_TBaseMeshObject(CL);
14484:   (FindClass('TOBJECT'),'TSkeletonFrameList
```

```
14485:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14486:   SIRegister_TSkeletonFrame(CL);
14487:   SIRegister_TSkeletonFrameList(CL);
14488:   (FindClass('TOBJECT'),'TSkeleton
14489:   (FindClass('TOBJECT'),'TSkeletonBone
14490:   SIRegister_TSkeletonBoneList(CL);
14491:   SIRegister_TSkeletonRootBoneList(CL);
14492:   SIRegister_TSkeletonBone(CL);
14493:   (FindClass('TOBJECT'),'TSkeletonColliderList
14494:   SIRegister_TSkeletonCollider(CL);
14495:   SIRegister_TSkeletonColliderList(CL);
14496:   (FindClass('TOBJECT'),'TGLBaseMesh
14497:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14498:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14499:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14500:    +'QuaternionList; end
14501:   SIRegister_TSkeleton(CL);
14502:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14503:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14504:   SIRegister_TMeshObject(CL);
14505:   SIRegister_TMeshObjectList(CL);
14506:   //TMeshObjectListClass', 'class of TMeshObjectList
14507:   (FindClass('TOBJECT'),'TMeshMorphTargetList
14508:   SIRegister_TMeshMorphTarget(CL);
14509:   SIRegister_TMeshMorphTargetList(CL);
14510:   SIRegister_TMorphableMeshObject(CL);
14511:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14512:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14513:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14514:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14515:   SIRegister_TSkeletonMeshObject(CL);
14516:   SIRegister_TFaceGroup(CL);
14517:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14518:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14519:   SIRegister_TFGVertexIndexList(CL);
14520:   SIRegister_TFGVertexNormalTexIndexList(CL);
14521:   SIRegister_TFGIndexTexCoordList(CL);
14522:   SIRegister_TFaceGroups(CL);
14523:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14524:   SIRegister_TVectorFile(CL);
14525:   //TVectorFileClass', 'class of TVectorFile
14526:   SIRegister_TGLGLSMVectorFile(CL);
14527:   SIRegister_TGLBaseMesh(CL);
14528:   SIRegister_TGLFreeForm(CL);
14529:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14530:   TGLActorOptions', 'set of TGLActorOption
14531:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14532:   (FindClass('TOBJECT'),'TGLActor
14533:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14534:   SIRegister_TActorAnimation(CL);
14535:   TActorAnimationName', 'String
14536:   SIRegister_TActorAnimations(CL);
14537:   SIRegister_TGLBaseAnimationControler(CL);
14538:   SIRegister_TGLAnimationControler(CL);
14539:   TActorFrameInterpolation', '( afpNone, afpLinear )
14540:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14541:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14542:   SIRegister_TGLActor(CL);
14543:   SIRegister_TVectorFileFormat(CL);
14544:   SIRegister_TVectorFileFormatsList(CL);
14545:   (FindClass('TOBJECT'),'EInvalidVectorFile
14546:  Function GetVectorFileFormats : TVectorFileFormatsList
14547:  Function VectorFileFormatsFilter : String
14548:  Function VectorFileFormatsSaveFilter : String
14549:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14550:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14551:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14552: end;
14553:
14554: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14555: begin
14556:  'Class_DColorPropPage','TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14557:  'Class_DFontPropPage','TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14558:  'Class_DPicturePropPage','TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14559:  'Class_DStringPropPage','TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14560:   SIRegister_TOleStream(CL);
14561:   (FindClass('TOBJECT'),'TConnectionPoints
14562:   TConnectionKind', '( ckSingle, ckMulti )
14563:   SIRegister_TConnectionPoint(CL);
14564:   SIRegister_TConnectionPoints(CL);
14565:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14566:   (FindClass('TOBJECT'),'TActiveXControlFactory
14567:   SIRegister_TActiveXControl(CL);
14568:   //TActiveXControlClass', 'class of TActiveXControl
14569:   SIRegister_TActiveXControlFactory(CL);
14570:   SIRegister_TActiveFormControl(CL);
14571:   SIRegister_TActiveForm(CL);
14572:   //TActiveFormClass', 'class of TActiveForm
14573:   SIRegister_TActiveFormFactory(CL);
```

```
14574:   (FindClass('TOBJECT'),'TPropertyPageImpl
14575:   SIRegister_TPropertyPage(CL);
14576:   //TPropertyPageClass', 'class of TPropertyPage
14577:   SIRegister_TPropertyPageImpl(CL);
14578:   SIRegister_TActiveXPropertyPage(CL);
14579:   SIRegister_TActiveXPropertyPageFactory(CL);
14580:   SIRegister_TCustomAdapter(CL);
14581:   SIRegister_TAdapterNotifier(CL);
14582:   SIRegister_IFontAccess(CL);
14583:   SIRegister_TFontAdapter(CL);
14584:   SIRegister_IPictureAccess(CL);
14585:   SIRegister_TPictureAdapter(CL);
14586:   SIRegister_TOleGraphic(CL);
14587:   SIRegister_TStringsAdapter(CL);
14588:   SIRegister_TReflectorWindow(CL);
14589:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
14590:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14591:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14592:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14593:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14594:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14595:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14596:  Function ParkingWindow : HWND
14597: end;
14598:
14599: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14600: begin
14601:  // TIp6Bytes = array [0..15] of Byte;
14602: {:binary form of IPv6 adress (for string conversion routines)}
14603:  // TIp6Words = array [0..7] of Word;
14604:   AddTypeS('TIp6Bytes', 'array [0..15] of Byte;');
14605:   AddTypeS('TIp6Words', 'array [0..7] of Word;');
14606:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14607:  Function synaIsIP( const Value : string) : Boolean');
14608:  Function synaIsIP6( const Value : string) : Boolean');
14609:  Function synaIPToID( Host : string) : Ansistring');
14610:  Function synaStrToIp6( value : string) : TIp6Bytes');
14611:  Function synaIp6ToStr( value : TIp6Bytes) : string');
14612:  Function synaStrToIp( value : string) : integer');
14613:  Function synaIpToStr( value : integer) : string');
14614:  Function synaReverseIP( Value : AnsiString) : AnsiString');
14615:  Function synaReverseIP6( Value : AnsiString) : AnsiString');
14616:  Function synaExpandIP6( Value : AnsiString) : AnsiString');
14617:  Function xStrToIP( const Value : String) : TIPAdr');
14618:  Function xIPToStr( const Adresse : TIPAdr) : String');
14619:  Function IPToCardinal( const Adresse : TIPAdr) : Cardinal');
14620:  Function CardinalToIP( const Value : Cardinal) : TIPAdr');
14621: end;
14622:
14623: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14624: begin
14625:  AddTypeS('TSpecials', 'set of Char');
14626:  Const('SpecialChar','TSpecials').SetSet( '=()[]<>:;,@/?\"_');
14627:  Const('URLFullSpecialChar','TSpecials').SetSet( ';/?:@=&#+');
14628:  Const('TableBase64'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14629:  Const('TableBase64mod'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14630:  Const('TableUU'( !"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14631:  Const('TableXX'(+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14632:  Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString');
14633:  Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14634:  Function DecodeURL( const Value : AnsiString) : AnsiString');
14635:  Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString');
14636:  Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString');
14637:  Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString');
14638:  Function EncodeURLElement( const Value : AnsiString) : AnsiString');
14639:  Function EncodeURL( const Value : AnsiString) : AnsiString');
14640:  Function Decode4to3( const Value, Table : AnsiString) : AnsiString');
14641:  Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString');
14642:  Function Encode3to4( const Value, Table : AnsiString) : AnsiString');
14643:  Function synDecodeBase64( const Value : AnsiString) : AnsiString');
14644:  Function synEncodeBase64( const Value : AnsiString) : AnsiString');
14645:  Function DecodeBase64mod( const Value : AnsiString) : AnsiString');
14646:  Function EncodeBase64mod( const Value : AnsiString) : AnsiString');
14647:  Function DecodeUU( const Value : AnsiString) : AnsiString');
14648:  Function EncodeUU( const Value : AnsiString) : AnsiString');
14649:  Function DecodeXX( const Value : AnsiString) : AnsiString');
14650:  Function DecodeYEnc( const Value : AnsiString) : AnsiString');
14651:  Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer');
14652:  Function synCrc32( const Value : AnsiString) : Integer');
14653:  Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word');
14654:  Function Crc16( const Value : AnsiString) : Word');
14655:  Function synMD5( const Value : AnsiString) : AnsiString');
14656:  Function HMAC_MD5( Text, Key : AnsiString) : AnsiString');
14657:  Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14658:  Function synSHA1( const Value : AnsiString) : AnsiString');
14659:  Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString');
14660:  Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14661:  Function synMD4( const Value : AnsiString) : AnsiString');
14662: end;
```

```
14663:
14664: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14665: begin
14666:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14667:    +'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14668:    +'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14669:    +'4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14670:    +'_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14671:    +' C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14672:    +', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14673:    +', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14674:    +'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14675:    +'1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14676:    +'2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14677:    +'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14678:    +'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14679:    +', CP864, CP865, CP869, CP1125 )');
14680:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14681:  Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14682:  Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
       TransformTable : array of Word) : AnsiString');
14683:  Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
       TransformTable : array of Word; Translit : Boolean) : AnsiString');
14684:  Function GetCurCP : TMimeChar');
14685:  Function GetCurOEMCP : TMimeChar');
14686:  Function GetCPFromID( Value : AnsiString) : TMimeChar');
14687:  Function GetIDFromCP( Value : TMimeChar) : AnsiString');
14688:  Function NeedCharsetConversion( const Value : AnsiString) : Boolean');
14689:  Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14690:  Function GetBOM( Value : TMimeChar) : AnsiString');
14691:  Function StringToWide( const Value : AnsiString) : WideString');
14692:  Function WideToString( const Value : WideString) : AnsiString');
14693: end;
14694:
14695: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14696: begin
14697:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14698:  Procedure WakeOnLan( MAC, IP : string)');
14699:  Function GetDNS : string');
14700:  Function GetIEProxy( protocol : string) : TProxySetting');
14701:  Function GetLocalIPs : string');
14702: end;
14703:
14704:
14705: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14706: begin
14707:  AddConstantN('synCR','Char #$0d);
14708:  Const('synLF','Char #$0a);
14709:  Const('cSerialChunk','LongInt'( 8192);
14710:  Const('LockfileDirectory','String '/var/lock');
14711:  Const('PortIsClosed','LongInt'( - 1);
14712:  Const('ErrAlreadyOwned','LongInt'( 9991);
14713:  Const('ErrAlreadyInUse','LongInt'( 9992);
14714:  Const('ErrWrongParameter','LongInt'( 9993);
14715:  Const('ErrPortNotOpen','LongInt'( 9994);
14716:  Const('ErrNoDeviceAnswer','LongInt'( 9995);
14717:  Const('ErrMaxBuffer','LongInt'( 9996);
14718:  Const('ErrTimeout','LongInt'( 9997);
14719:  Const('ErrNotRead','LongInt'( 9998);
14720:  Const('ErrFrame','LongInt'( 9999);
14721:  Const('ErrOverrun','LongInt'( 10000);
14722:  Const('ErrRxOver','LongInt'( 10001);
14723:  Const('ErrRxParity','LongInt'( 10002);
14724:  Const('ErrTxFull','LongInt'( 10003);
14725:  Const('dcb_Binary','LongWord')( $00000001);
14726:  Const('dcb_ParityCheck','LongWord')( $00000002);
14727:  Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14728:  Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14729:  Const('dcb_DtrControlMask','LongWord')( $00000030);
14730:  Const('dcb_DtrControlDisable','LongWord')( $00000000);
14731:  Const('dcb_DtrControlEnable','LongWord')( $00000010);
14732:  Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14733:  Const('dcb_DsrSensivity','LongWord')( $00000040);
14734:  Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14735:  Const('dcb_OutX','LongWord')( $00000100);
14736:  Const('dcb_InX','LongWord')( $00000200);
14737:  Const('dcb_ErrorChar','LongWord')( $00000400);
14738:  Const('dcb_NullStrip','LongWord')( $00000800);
14739:  Const('dcb_RtsControlMask','LongWord')( $00003000);
14740:  Const('dcb_RtsControlDisable','LongWord')( $00000000);
14741:  Const('dcb_RtsControlEnable','LongWord')( $00001000);
14742:  Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14743:  Const('dcb_RtsControlToggle','LongWord')( $00003000);
14744:  Const('dcb_AbortOnError','LongWord')( $00004000);
14745:  Const('dcb_Reserveds','LongWord')( $FFFF8000);
14746:  Const('synSB1','LongInt'( 0);
14747:  Const('SB1andHalf','LongInt'( 1);
14748:  Const('synSB2','LongInt'( 2);
14749:  Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 ));
```

```
14750:  Const('CS7fix','LongWord')( $0000020);
14751:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14752:    +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14753:    +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14754:    +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14755:   //AddTypeS('PDCB', '^TDCB // will not work');
14756:  //Const('MaxRates','LongInt'( 18);
14757:  //Const('MaxRates','LongInt'( 30);
14758:  //Const('MaxRates','LongInt'( 19);
14759:  Const('O_SYNC','LongWord')( $0080);
14760:  Const('synOK','LongInt'( 0);
14761:  Const('synErr','LongInt'( integer ( - 1 ));
14762:   AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
      HR_WriteCount, HR_Wait )');
14763:  Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string)');
14764:   SIRegister_ESynaSerError(CL);
14765:   SIRegister_TBlockSerial(CL);
14766:  Function GetSerialPortNames : string');
14767: end;
14768:
14769: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14770: begin
14771:  Const('DLLIconvName','String 'libiconv.so');
14772:  Const('DLLIconvName','String 'iconv.dll');
14773:   AddTypeS('size_t', 'Cardinal');
14774:   AddTypeS('iconv_t', 'Integer');
14775:   //AddTypeS('iconv_t', 'Pointer');
14776:   AddTypeS('argptr', 'iconv_t');
14777:  Function SynaIconvOpen( const tocode, fromcode : Ansistring) : iconv_t');
14778:  Function SynaIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t');
14779:  Function SynaIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t');
14780:  Function SynaIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer');
14781:  Function SynaIconvClose( var cd : iconv_t) : integer');
14782:  Function SynaIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer');
14783:  Function IsIconvloaded : Boolean');
14784:  Function InitIconvInterface : Boolean');
14785:  Function DestroyIconvInterface : Boolean');
14786:  Const('ICONV_TRIVIALP','LongInt'( 0);
14787:  Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14788:  Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14789:  Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14790:  Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14791: end;
14792:
14793: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14794: begin
14795:  Const 'ICMP_ECHO','LongInt'( 8);
14796:  Const('ICMP_ECHOREPLY','LongInt'( 0);
14797:  Const('ICMP_UNREACH','LongInt'( 3);
14798:  Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14799:  Const('ICMP6_ECHO','LongInt'( 128);
14800:  Const('ICMP6_ECHOREPLY','LongInt'( 129);
14801:  Const('ICMP6_UNREACH','LongInt'( 1);
14802:  Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14803:   AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14804:    +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14805:   SIRegister_TPINGSend(CL);
14806:  Function PingHost( const Host : string) : Integer');
14807:  Function TraceRouteHost( const Host : string) : string');
14808: end;
14809:
14810: procedure SIRegister_asn1util(CL: TPSPascalCompiler);
14811: begin
14812:  AddConstantN('synASN1_BOOL','LongWord')( $01);
14813:  Const('synASN1_INT','LongWord')( $02);
14814:  Const('synASN1_OCTSTR','LongWord')( $04);
14815:  Const('synASN1_NULL','LongWord')( $05);
14816:  Const('synASN1_OBJID','LongWord')( $06);
14817:  Const('synASN1_ENUM','LongWord')( $0a);
14818:  Const('synASN1_SEQ','LongWord')( $30);
14819:  Const('synASN1_SETOF','LongWord')( $31);
14820:  Const('synASN1_IPADDR','LongWord')( $40);
14821:  Const('synASN1_COUNTER','LongWord')( $41);
14822:  Const('synASN1_GAUGE','LongWord')( $42);
14823:  Const('synASN1_TIMETICKS','LongWord')( $43);
14824:  Const('synASN1_OPAQUE','LongWord')( $44);
14825:  Function synASNEncOIDItem( Value : Integer) : AnsiString');
14826:  Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer');
14827:  Function synASNEncLen( Len : Integer) : AnsiString');
14828:  Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer');
14829:  Function synASNEncInt( Value : Integer) : AnsiString');
14830:  Function synASNEncUInt( Value : Integer) : AnsiString');
14831:  Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString');
14832:  Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14833:  Function synMibToId( Mib : String) : AnsiString');
14834:  Function synIdToMib( const Id : AnsiString) : String');
14835:  Function synIntMibToStr( const Value : AnsiString) : AnsiString');
14836:  Function ASNdump( const Value : AnsiString) : AnsiString');
14837:  Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean');
```

```
14838:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14839: end;
14840:
14841: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14842: begin
14843:  Const('cLDAPProtocol','String '389');
14844:  Const('LDAP_ASN1_BIND_REQUEST','LongWord')( $60);
14845:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord')( $61);
14846:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord')( $42);
14847:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord')( $63);
14848:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord')( $64);
14849:  Const('LDAP_ASN1_SEARCH_DONE','LongWord')( $65);
14850:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord')( $73);
14851:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord')( $66);
14852:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord')( $67);
14853:  Const('LDAP_ASN1_ADD_REQUEST','LongWord')( $68);
14854:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord')( $69);
14855:  Const('LDAP_ASN1_DEL_REQUEST','LongWord')( $4A);
14856:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord')( $6B);
14857:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord')( $6C);
14858:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord')( $6D);
14859:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord')( $6E);
14860:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord')( $6F);
14861:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord')( $70);
14862:  Const('LDAP_ASN1_EXT_REQUEST','LongWord')( $77);
14863:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord')( $78);
14864:   SIRegister_TLDAPAttribute(CL);
14865:   SIRegister_TLDAPAttributeList(CL);
14866:   SIRegister_TLDAPResult(CL);
14867:   SIRegister_TLDAPResultList(CL);
14868:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14869:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14870:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14871:   SIRegister_TLDAPSend(CL);
14872:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString');
14873: end;
14874:
14875:
14876: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14877: begin
14878:  Const('cSysLogProtocol','String '514');
14879:  Const('FCL_Kernel','LongInt'( 0);
14880:  Const('FCL_UserLevel','LongInt'( 1);
14881:  Const('FCL_MailSystem','LongInt'( 2);
14882:  Const('FCL_System','LongInt'( 3);
14883:  Const('FCL_Security','LongInt'( 4);
14884:  Const('FCL_Syslogd','LongInt'( 5);
14885:  Const('FCL_Printer','LongInt'( 6);
14886:  Const('FCL_News','LongInt'( 7);
14887:  Const('FCL_UUCP','LongInt'( 8);
14888:  Const('FCL_Clock','LongInt'( 9);
14889:  Const('FCL_Authorization','LongInt'( 10);
14890:  Const('FCL_FTP','LongInt'( 11);
14891:  Const('FCL_NTP','LongInt'( 12);
14892:  Const('FCL_LogAudit','LongInt'( 13);
14893:  Const('FCL_LogAlert','LongInt'( 14);
14894:  Const('FCL_Time','LongInt'( 15);
14895:  Const('FCL_Local0','LongInt'( 16);
14896:  Const('FCL_Local1','LongInt'( 17);
14897:  Const('FCL_Local2','LongInt'( 18);
14898:  Const('FCL_Local3','LongInt'( 19);
14899:  Const('FCL_Local4','LongInt'( 20);
14900:  Const('FCL_Local5','LongInt'( 21);
14901:  Const('FCL_Local6','LongInt'( 22);
14902:  Const('FCL_Local7','LongInt'( 23);
14903:   Type(TSyslogSeverity','(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
14904:   SIRegister_TSyslogMessage(CL);
14905:   SIRegister_TSyslogSend(CL);
14906:  Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
      Content:string):Boolean;
14907: end;
14908:
14909:
14910: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14911: begin
14912:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14913:   SIRegister_TMessHeader(CL);
14914:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14915:   SIRegister_TMimeMess(CL);
14916: end;
14917:
14918: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14919: begin
14920:   (FindClass('TOBJECT'),'TMimePart');
14921:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
14922:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14923:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14924:   SIRegister_TMimePart(CL);
14925:  Const('MaxMimeType','LongInt'( 25);
```

```
14926:  Function GenerateBoundary : string');
14927: end;
14928:
14929: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14930: begin
14931:  Function InlineDecode( const Value : string; CP : TMimeChar) : string');
14932:  Function InlineEncode( const Value : string; CP, MimeP : TMimeChar) : string');
14933:  Function NeedInline( const Value : AnsiString) : boolean');
14934:  Function InlineCodeEx( const Value : string; FromCP : TMimeChar) : string');
14935:  Function InlineCode( const Value : string) : string');
14936:  Function InlineEmailEx( const Value : string; FromCP : TMimeChar) : string');
14937:  Function InlineEmail( const Value : string) : string');
14938: end;
14939:
14940: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
14941: begin
14942:  Const('cFtpProtocol','String '21');
14943:  Const('cFtpDataProtocol','String '20');
14944:  Const('FTP_OK','LongInt'( 255);
14945:  Const('FTP_ERR','LongInt'( 254);
14946:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
14947:   SIRegister_TFTPListRec(CL);
14948:   SIRegister_TFTPList(CL);
14949:   SIRegister_TFTPSend(CL);
14950:  Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
14951:  Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string) : Boolean');
14952:  Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
      ToPort, ToFile, ToUser, ToPass : string) : Boolean');
14953: end;
14954:
14955: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
14956: begin
14957:  Const('cHttpProtocol','String '80');
14958:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
14959:   SIRegister_THTTPSend(CL);
14960:  Function HttpGetText( const URL : string; const Response : TStrings) : Boolean');
14961:  Function HttpGetBinary( const URL : string; const Response : TStream) : Boolean');
14962:  Function HttpPostBinary( const URL : string; const Data : TStream) : Boolean');
14963:  Function HttpPostURL( const URL, URLData : string; const Data : TStream) : Boolean');
14964:  Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
      ResultData:TStrings):Bool;
14965: end;
14966:
14967: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
14968: begin
14969:  Const('cSmtpProtocol','String '25');
14970:   SIRegister_TSMTPSend(CL);
14971:  Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
      Passw:string):Bool;
14972:  Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
14973:  Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
      Username, Password : string):Boolean');
14974: end;
14975:
14976: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
14977: begin
14978:  Const('cSnmpProtocol','String '161');
14979:  Const('cSnmpTrapProtocol','String '162');
14980:  Const('SNMP_V1','LongInt'( 0);
14981:  Const('SNMP_V2C','LongInt'( 1);
14982:  Const('SNMP_V3','LongInt'( 3);
14983:  Const('PDUGetRequest','LongWord')( $A0);
14984:  Const('PDUGetNextRequest','LongWord')( $A1);
14985:  Const('PDUGetResponse','LongWord')( $A2);
14986:  Const('PDUSetRequest','LongWord')( $A3);
14987:  Const('PDUTrap','LongWord')( $A4);
14988:  Const('PDUGetBulkRequest','LongWord')( $A5);
14989:  Const('PDUInformRequest','LongWord')( $A6);
14990:  Const('PDUTrapV2','LongWord')( $A7);
14991:  Const('PDUReport','LongWord')( $A8);
14992:  Const('ENoError',LongInt 0;
14993:  Const('ETooBig','LongInt')( 1);
14994:  Const('ENoSuchName','LongInt'( 2);
14995:  Const('EBadValue','LongInt'( 3);
14996:  Const('EReadOnly','LongInt'( 4);
14997:  Const('EGenErr','LongInt'( 5);
14998:  Const('ENoAccess','LongInt'( 6);
14999:  Const('EWrongType','LongInt'( 7);
15000:  Const('EWrongLength','LongInt'( 8);
15001:  Const('EWrongEncoding','LongInt'( 9);
15002:  Const('EWrongValue','LongInt'( 10);
15003:  Const('ENoCreation','LongInt'( 11);
15004:  Const('EInconsistentValue','LongInt'( 12);
15005:  Const('EResourceUnavailable','LongInt'( 13);
15006:  Const('ECommitFailed','LongInt'( 14);
15007:  Const('EUndoFailed','LongInt'( 15);
15008:  Const('EAuthorizationError','LongInt'( 16);
15009:  Const('ENotWritable','LongInt'( 17);
15010:  Const('EInconsistentName','LongInt'( 18);
```

```
15011:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15012:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15013:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15014:   SIRegister_TSNMPMib(CL);
15015:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15016:    +'EngineTime : integer; EngineStamp : Cardinal; end');
15017:   SIRegister_TSNMPRec(CL);
15018:   SIRegister_TSNMPSend(CL);
15019:  Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean');
15020:  Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean');
15021:  Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15022:  Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15023:  Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15024:  Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
        Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer');
15025:  Function RecvTrap(var Dest,Source,Enterprise,Community: AnsiString; var Generic, Specific, Seconds :
        Integer; const MIBName, MIBValue : TStringList) : Integer');
15026: end;
15027:
15028: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15029: begin
15030:  Function GetDomainName2: AnsiString');
15031:  Function GetDomainController( Domain : AnsiString) : AnsiString');
15032:  Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15033:  Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15034:  Function GetDateTime( Controller : AnsiString) : TDateTime');
15035:  Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15036: end;
15037:
15038: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15039: begin
15040:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15041:   TwwDateTimeSelection','(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15042:  Function wwStrToDate( const S : string) : boolean');
15043:  Function wwStrToTime( const S : string) : boolean');
15044:  Function wwStrToDateTime( const S : string) : boolean');
15045:  Function wwStrToTimeVal( const S : string) : TDateTime');
15046:  Function wwStrToDateVal( const S : string) : TDateTime');
15047:  Function wwStrToDateTimeVal( const S : string) : TDateTime');
15048:  Function wwStrToInt( const S : string) : boolean');
15049:  Function wwStrToFloat( const S : string) : boolean');
15050:  Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15051:  Function wwNextDay( Year, Month, Day : Word) : integer');
15052:  Function wwPriorDay( Year, Month, Day : Word) : integer');
15053:  Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15054:  Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15055:  Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15056:  Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15057:  Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15058:  Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15059:  Procedure wwSetDateTimeCursorSelection(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15060:  Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15061: end;
15062:
15063: unit uPSI_Themes;
15064: Function ThemeServices : TThemeServices');
15065: Function ThemeControl( AControl : TControl) : Boolean');
15066: Function UnthemedDesigner( AControl : TControl) : Boolean');
15067: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15068: begin
15069:  Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15070: end;
15071: Unit uPSC_menus;
15072:  Function StripHotkey( const Text : string) : string');
15073:  Function GetHotkey( const Text : string) : string');
15074:  Function AnsiSameCaption( const Text1, Text2 : string) : Boolean');
15075:  Function IsAltGRPressed : boolean');
15076:
15077: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15078: begin
15079:   TCommandEvent','Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15080:   SIRegister_TIdIMAP4Server(CL);
15081: end;
15082:
15083: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15084: begin
15085:  'HASH_SIZE','LongInt'( 256);
15086:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15087:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15088:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15089:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15090:    +' : Integer; Value : Variant; end');
15091:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15092:   SIRegister_TVariantSymbolTable(CL);
15093: end;
15094:
15095: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15096: begin
15097:   SIRegister_TThreadLocalVariables(CL);
```

```
15098:  Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD) : PChar');
15099:  //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD) : PISC_QUAD');
15100:  Function ThreadLocals : TThreadLocalVariables');
15101:  Procedure WriteDebug( sz : String)');
15102:  CL.AddConstantN('UDF_SUCCESS','LongInt'( 0);
15103:  'UDF_FAILURE','LongInt'( 1);
15104:  'cSignificantlyLarger','LongInt'( 1024 * 4);
15105:  CL.AddTypeS('mTByteArray', 'array of byte;');
15106:  function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15107:  function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15108:  procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTarget: string);
15109:  function IsNetworkConnected: Boolean;
15110:  function IsInternetConnected: Boolean;
15111:  function IsCOMConnected: Boolean;
15112:  function IsNetworkOn: Boolean;
15113:  function IsInternetOn: Boolean;
15114:  function IsCOMOn: Boolean;
15115:  Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15116:  Function KillTimer( hWnd : HWND; uIDEvent : UINT) : BOOL');
15117:  Function wIsWindowUnicode( hWnd : HWND) : BOOL');
15118:  Function wEnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL');
15119:  Function wIsWindowEnabled( hWnd : HWND) : BOOL');
15120:  Function GetMenu( hWnd : HWND) : HMENU');
15121:  Function SetMenu( hWnd : HWND; hMenu : HMENU) : BOOL');
15122: end;
15123:
15124: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15125: begin
15126:   SIRegister_IDataBlock(CL);
15127:   SIRegister_ISendDataBlock(CL);
15128:   SIRegister_ITransport(CL);
15129:   SIRegister_TDataBlock(CL);
15130:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15131:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15132:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15133:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15134:   SIRegister_TCustomDataBlockInterpreter(CL);
15135:   SIRegister_TSendDataBlock(CL);
15136:   'CallSig','LongWord')( $D800);
15137:   'ResultSig','LongWord')( $D400);
15138:   'asMask','LongWord')( $00FF);
15139:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15140:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15141:  Procedure CheckSignature( Sig : Integer)');
15142: end;
15143:
15144: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15145: begin
15146:   //CL.AddTypeS('HINTERNET', '___Pointer');
15147:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15148:   CL.AddTypeS('HINTERNET', 'Integer');
15149:   CL.AddTypeS('HINTERNET2', '___Pointer');
15150:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15151:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15152:   CL.AddTypeS('INTERNET_PORT', 'Word');
15153:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15154:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15155:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15156:  'INTERNET_RFC1123_FORMAT','LongInt'( 0);
15157:  'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15158:  Function InternetCrackUrl(lpszUrl:PChar; dwUrlLength,dwFlags:DWORD; var
       lpUrlComponents:TURLComponents):BOOL;
15159:  Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
       dwUrlLength:DWORD):BOOL;
15160:  Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15161:  Function
       InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
       lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15162:  Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
15163:  ;dwContext:DWORD):HINTERNET;
15164:  Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
       lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15165:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
       dwContext:DWORD):BOOL;
15166:  Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE) : BOOL');
15167:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15168:  Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD) : DWORD');
15169:  Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD) : BOOL');
15170:  Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15171:  Function InternetAutodialHangup( dwReserved : DWORD) : BOOL');
15172:  Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD) : BOOL');
15173:  Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
       lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15174:  Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
       lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET) : BOOL');
15175:  Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15176:  ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15177:  Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
       dwContext:DWORD):BOOL;
```

```
15178:  Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
       TWin32FindData; dwFlags : DWORD; dwContext : DWORD) : HINTERNET');
15179:  Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
       BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD) : BOOL');
15180:  Function
       WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15181:  Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar) : BOOL');
15182:  Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15183:  Function
       FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15184:  Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15185:  Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar) : BOOL');
15186:  Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar) : BOOL');
15187:  Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15188:  Function
       FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15189:  Function IS_GOPHER_FILE( GopherType : DWORD) : BOOL');
15190:  Function IS_GOPHER_DIRECTORY( GopherType : DWORD) : BOOL');
15191:  Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD) : BOOL');
15192:  Function IS_GOPHER_ERROR( GopherType : DWORD) : BOOL');
15193:  Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD) : BOOL');
15194:  Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD) : BOOL');
15195:  Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD) : BOOL');
15196:  Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD) : BOOL');
15197:  Function IS_GOPHER_ASK( GopherType : DWORD) : BOOL');
15198:  Function IS_GOPHER_PLUS( GopherType : DWORD) : BOOL');
15199:  Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD) : BOOL');
15200:  Function
       GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
       PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15201:  Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL');
15202:  Function
       GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15203:  Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
       PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15204:  Function
       HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15205:  Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
       dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15206:  Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15207:  Function InternetAttemptConnect( dwReserved : DWORD) : DWORD');
15208:  Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD) : BOOL');
15209:  Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15210:  Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL) : DWORD');
15211:  Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject) : Int64');
15212:  Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject) : Bool');
15213:  Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
       lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15214:  Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
       lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15215:  Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15216:  Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15217:  Function
       InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15218:  Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD) : BOOL');
15219: end;
15220:
15221: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15222: begin
15223:   AddTypeS('strCharSet', 'set of char');
15224:   TwwGetWordOption','(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15225:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15226:  Procedure strBreakApart( s : string; delimeter : string; parts : TStrings)');
15227:  Function strGetToken( s : string; delimeter : string; var APos : integer) : string');
15228:  Procedure strStripPreceding( var s : string; delimeter : strCharSet)');
15229:  Procedure strStripTrailing( var s : string; delimeter : strCharSet)');
15230:  Procedure strStripWhiteSpace( var s : string)');
15231:  Function strRemoveChar( str : string; removeChar : char) : string');
15232:  Function wwstrReplaceChar( str : string; removeChar, replaceChar : char) : string');
15233:  Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string');
15234:  Function wwEqualStr( s1, s2 : string) : boolean');
15235:  Function strCount( s : string; delimeter : char) : integer');
15236:  Function strWhiteSpace : strCharSet');
15237:  Function wwExtractFileNameOnly( const FileName : string) : string');
15238:  Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15239:  Function strTrailing( s : string; delimeter : char) : string');
15240:  Function strPreceding( s : string; delimeter : char) : string');
15241:  Function wwstrReplace( s, Find, Replace : string) : string');
15242: end;
15243:
15244: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15245: begin
15246:   SIRegister_TRemoteDataModule(CL);
15247:   SIRegister_TCRemoteDataModule(CL);
15248:  Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)');
15249:  Procedure UnregisterPooled( const ClassID : string)');
15250:  Procedure EnableSocketTransport( const ClassID : string)');
15251:  Procedure DisableSocketTransport( const ClassID : string)');
15252:  Procedure EnableWebTransport( const ClassID : string)');
```

```
15253:  Procedure DisableWebTransport( const ClassID : string)');
15254: end;
15255:
15256: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15257: begin
15258:  Function mxArcCos( x : Real) : Real');
15259:  Function mxArcSin( x : Real) : Real');
15260:  Function Comp2Str( N : Comp) : String');
15261:  Function Int2StrPad0( N : LongInt; Len : Integer) : String');
15262:  Function Int2Str( N : LongInt) : String');
15263:  Function mxIsEqual( R1, R2 : Double) : Boolean');
15264:  Function LogXY( x, y : Real) : Real');
15265:  Function Pennies2Dollars( C : Comp) : String');
15266:  Function mxPower( X : Integer; Y : Integer) : Real');
15267:  Function Real2Str( N : Real; Width, Places : integer) : String');
15268:  Function mxStr2Comp( MyString : string) : Comp');
15269:  Function mxStr2Pennies( S : String) : Comp');
15270:  Function Str2Real( MyString : string) : Real');
15271:  Function XToTheY( x, y : Real) : Real');
15272: end;
15273:
15274: //************************Cindy Functions!*****************************
15275: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15276: begin
15277:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15278:    +'_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15279:    +'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15280:  MessagePlainText','String 'text/plain');
15281:  CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15282:  MessageAlterText_Html','String 'multipart/alternative');
15283:  MessageHtml_Attach','String 'multipart/mixed');
15284:  MessageHtml_RelatedAttach','String 'multipart/related; type="text/html"');
15285:  MessageAlterText_Html_Attach','String 'multipart/mixed');
15286:  MessageAlterText_Html_RelatedAttach','String')('multipart/related;type="multipart/alternative"';
15287:  MessageAlterText_Html_Attach_RelatedAttach','String 'multipart/mixed');
15288:  MessageReadNotification','String').( 'multipart/report; report-type="disposition-notification"';
15289:  Function ForceDecodeHeader( aHeader : String) : String');
15290:  Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string');
15291:  Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;');
15292:  Function Base64_DecodeToBytes( Value : String) : TBytes;');
15293:  Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15294:  Function Get_MD5( const aFileName : string) : string');
15295:  Function Get_MD5FromString( const aString : string) : string');
15296: end;
15297:
15298: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15299: begin
15300:  Function IsFolder( SRec : TSearchrec) : Boolean');
15301:  Function isFolderReadOnly( Directory : String) : Boolean');
15302:  Function DirectoryIsEmpty( Directory : String) : Boolean');
15303:  Function DirectoryWithSubDir( Directory : String) : Boolean');
15304:  Procedure GetSubDirs( FromDirectory : String; aList : TStrings)');
15305:  Function DiskFreeBytes( Drv : Char) : Int64');
15306:  Function DiskBytes( Drv : Char) : Int64');
15307:  Function GetFileBytes( Filename : String) : Int64');
15308:  Function GetFilesBytes( Directory, Filter : String) : Int64');
15309:  SE_CREATE_TOKEN_NAME','String 'SeCreateTokenPrivilege');
15310:  SE_ASSIGNPRIMARYTOKEN_NAME','String 'SeAssignPrimaryTokenPrivilege');
15311:  SE_LOCK_MEMORY_NAME','String 'SeLockMemoryPrivilege');
15312:  SE_INCREASE_QUOTA_NAME','String 'SeIncreaseQuotaPrivilege');
15313:  SE_UNSOLICITED_INPUT_NAME','String 'SeUnsolicitedInputPrivilege');
15314:  SE_MACHINE_ACCOUNT_NAME','String 'SeMachineAccountPrivilege');
15315:  SE_TCB_NAME','String 'SeTcbPrivilege');
15316:  SE_SECURITY_NAME','String 'SeSecurityPrivilege');
15317:  SE_TAKE_OWNERSHIP_NAME','String 'SeTakeOwnershipPrivilege');
15318:  SE_LOAD_DRIVER_NAME','String 'SeLoadDriverPrivilege');
15319:  SE_SYSTEM_PROFILE_NAME','String 'SeSystemProfilePrivilege');
15320:  SE_SYSTEMTIME_NAME','String 'SeSystemtimePrivilege');
15321:  SE_PROF_SINGLE_PROCESS_NAME','String 'SeProfileSingleProcessPrivilege');
15322:  SE_INC_BASE_PRIORITY_NAME','String 'SeIncreaseBasePriorityPrivilege');
15323:  SE_CREATE_PAGEFILE_NAME','String 'SeCreatePagefilePrivilege');
15324:  SE_CREATE_PERMANENT_NAME','String 'SeCreatePermanentPrivilege');
15325:  SE_BACKUP_NAME','String 'SeBackupPrivilege');
15326:  SE_RESTORE_NAME','String 'SeRestorePrivilege');
15327:  SE_SHUTDOWN_NAME','String 'SeShutdownPrivilege');
15328:  SE_DEBUG_NAME','String 'SeDebugPrivilege');
15329:  SE_AUDIT_NAME','String 'SeAuditPrivilege');
15330:  SE_SYSTEM_ENVIRONMENT_NAME','String 'SeSystemEnvironmentPrivilege');
15331:  SE_CHANGE_NOTIFY_NAME','String 'SeChangeNotifyPrivilege');
15332:  SE_REMOTE_SHUTDOWN_NAME','String 'SeRemoteShutdownPrivilege');
15333:  SE_UNDOCK_NAME','String 'SeUndockPrivilege');
15334:  SE_SYNC_AGENT_NAME','String 'SeSyncAgentPrivilege');
15335:  SE_ENABLE_DELEGATION_NAME','String 'SeEnableDelegationPrivilege');
15336:  SE_MANAGE_VOLUME_NAME','String 'SeManageVolumePrivilege');
15337: end;
15338:
15339:
15340: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15341: begin
```

```
15342:   Type(TWindowsVersion', '( wvUnknown, wvWin31, wvWin95, wvWin98, wvWin'
15343:    +'Me, wvWinNt3, wvWinNt4, wvWin2000, wvWinXP, wvWinVista, wvWin7, wvWin8, wvWin8_Or_Upper )');
15344:  Function ShellGetExtensionName( FileName : String) : String');
15345:  Function ShellGetIconIndex( FileName : String) : Integer');
15346:  Function ShellGetIconHandle( FileName : String) : HIcon');
15347:  Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)');
15348:  Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)');
15349:  Procedure ShellRenameDir( DirFrom, DirTo : string)');
15350:  Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15351:  Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15352:  Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)');
15353:  Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)');
15354:  Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
       WaitCloseCompletion : boolean) : Boolean');
15355:  Procedure RestoreAndSetForegroundWindow( Hnd : Integer)');
15356:  Function RemoveDuplicatedPathDelimiter( Str : String) : String');
15357:  Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime');
15358:  Function GetModificationDate( Filename : String) : TDateTime');
15359:  Function GetCreationDate( Filename : String) : TDateTime');
15360:  Function GetLastAccessDate( Filename : String) : TDateTime');
15361:  Function FileDelete( Filename : String) : Boolean');
15362:  Function FileIsOpen( Filename : string) : boolean');
15363:  Procedure FilesDelete( FromDirectory : String; Filter : ShortString)');
15364:  Function DirectoryDelete( Directory : String) : Boolean');
15365:  Function GetPrinters( PrintersList : TStrings) : Integer');
15366:  Procedure SetDefaultPrinter( PrinterName : String)');
15367:  Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)');
15368:  Function WinToDosPath( WinPathName : String) : String');
15369:  Function DosToWinPath( DosPathName : String) : String');
15370:  Function cyGetWindowsVersion : TWindowsVersion');
15371:  Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean');
15372:  Procedure WindowsShutDown( Restart : boolean)');
15373:  Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
       FileIcon:string;NumIcone:integer);
15374:  Procedure GetWindowsFonts( FontsList : TStrings)');
15375:  Function GetAvailableFilename( DesiredFileName : String): String');
15376: end;
15377:
15378: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15379: begin
15380:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15381:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15382:   Type(TStringRead', '( srFromLeft, srFromRight )');
15383:   Type(TStringReads', 'set of TStringRead');
15384:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15385:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15386:   Type(TWordsOptions', 'set of TWordsOption');
15387:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15388:   Type(TCarTypes', 'set of TCarType');
15389:   //CL.AddTypeS('TUnicodeCategories', 'set of TUnicodeCategory');
15390:  CarTypeAlphabetic','LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15391:  Function Char_GetType( aChar : Char) : TCarType');
15392:  Function SubString_Count( Str : String; Separator : Char) : Integer');
15393:  Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15394:  Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String');
15395:  Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer');
15396:  Procedure SubString_Add( var Str : String; Separator : Char; Value : String)');
15397:  Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String)');
15398:  Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String)');
15399:  Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean');
15400:  Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15401:  Function SubString_Ribbon( Str : string; Separator : Char; Current: Word; MoveBy : Integer):Integer;');
15402:  Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String;');
15403:  Function String_Quote( Str : String) : String');
15404:  Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char');
15405:  Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15406:  Function String_GetWord( Str : String; StringRead : TStringRead) : String');
15407:  Function String_GetInteger( Str : String; StringRead : TStringRead) : String');
15408:  Function String_ToInt( Str : String) : Integer');
15409:  Function String_Uppercase( Str : String; Options : TWordsOptions) : String');
15410:  Function String_Lowercase( Str : String; Options : TWordsOptions) : String');
15411:  Function String_Reverse( Str : String) : String');
15412:  Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15413:  Function String_Pos1(SubStr:String; Str:String; StringRead:TStringRead; Occurrence:Word; CaseSensitive :
       TCaseSensitive):Integer;');
15414:  Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15415:  Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15416:  Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
       Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String;');
15417:  Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String;');
15418:  Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String;');
15419:  Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String');
15420:  Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String');
15421:  Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String');
15422:  Function String_End( Str : String; Cars : Word) : String');
15423:  Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
       AlwaysFindFromBeginning : Boolean) : String');
15424:  Function String_SubstCar( Str : String; Old, New : Char) : String');
15425:  Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer');
```

```
15426:  Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15427:  Function String_IsNumbers( Str : String) : Boolean');
15428:  Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer');
15429:  Function StringToCsvCell( aStr : String) : String');
15430: end;
15431:
15432: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15433: begin
15434:  Function LongDayName( aDate : TDate) : String');
15435:  Function LongMonthName( aDate : TDate) : String');
15436:  Function ShortYearOf( aDate : TDate) : byte');
15437:  Function DateToStrYYYYMMDD( aDate : TDate) : String');
15438:  Function StrYYYYMMDDToDate( aStr : String) : TDate');
15439:  Function SecondsToMinutes( Seconds : Integer) : Double');
15440:  Function MinutesToSeconds( Minutes : Double) : Integer');
15441:  Function MinutesToHours( Minutes : Integer) : Double');
15442:  Function HoursToMinutes( Hours : Double) : Integer');
15443:  Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime');
15444:  Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)');
15445:  Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime');
15446:  Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15447:  Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15448:  Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64;');
15449:  Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
       RsltBegin:TDateTime; RsltEnd : TDateTime) : Boolean;');
15450:  Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15451:  Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean');
15452: end;
15453:
15454: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15455: begin
15456:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15457:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15458:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15459:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15460:  Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer;');
15461:  Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer;');
15462:  Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer');
15463:  Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15464:  Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)');
15465:  Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode');
15466:  Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode');
15467:  Function
       TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode');
15468:  Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode');
15469:  Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
       CopySubChildren:Bool;
15470:  Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)');
15471:  Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15472:  Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl');
15473:  Procedure cyCenterControl( aControl : TControl)');
15474:  Function GetLastParent( aControl : TControl) : TWinControl');
15475:  Function GetControlBitmap( aControl : TWinControl) : TBitmap');
15476:  Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap');
15477: end;
15478:
15479: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15480: begin
15481:  Function TablePackTable( Tab : TTable) : Boolean');
15482:  Function TableRegenIndexes( Tab : TTable) : Boolean');
15483:  Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean');
15484:  Function TableUndeleteRecord( Tab : TTable) : Boolean');
15485:  Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15486:  Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean');
15487:  Function TableEmptyTable( Tab : TTable) : Boolean');
15488:  Function TableFindKey( aTable : TTable; Value : String) : Boolean');
15489:  Procedure TableFindNearest( aTable : TTable; Value : String)');
15490:  Function
       TableCreate(Owner:TComponent;DataBaseName:ShortString;TableName:String;IndexName:ShortString;ReadOnly :
       Boolean):TTable;
15491:  Function
       TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15492:  Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String');
15493: end;
15494:
15495: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15496: begin
15497:   SIRegister_TcyRunTimeDesign(CL);
15498:   SIRegister_TcyShadowText(CL);
15499:   SIRegister_TcyBgPicture(CL);
15500:   SIRegister_TcyGradient(CL);
15501:   SIRegister_tcyBevel(CL);
15502:   //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15503:   SIRegister_tcyBevels(CL);
15504:   SIRegister_TcyImagelistOptions(CL);
15505:  Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture)');
15506: end;
15507:
15508: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
```

```
15509: begin
15510:  Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
       adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
       Maxdegrade : Byte;'+
15511:   SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap)');
15512:  Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15513:  Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15514:  Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; toRect : TRect; OrientationShape : TDgradOrientationShape)');
15515:  Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
       Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap)');
15516:  Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer)');
15517:  Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer);');
15518:  Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
       BottomColor : TColor; const Width : Integer; const RoundRect : boolean);');
15519:  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;
15520:  Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15521:  Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
       aState : TButtonState; Focused, Hot : Boolean)');
15522:  Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
       TColor; aState : TButtonState; Focused, Hot : Boolean)');
15523:  Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
       GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean)');
15524:  Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
       const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor)');
15525:  Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
       TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer)');
15526:  Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
       TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15527:  Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
       WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;');
15528:  Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt)');
15529:  Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;');
15530:  Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;');
15531:  Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
       CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout)');
15532:  Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15533:  Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
       Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord');
15534:  Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean');
15535:  Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean');
15536:  Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean');
15537:  Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean');
15538:  Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect);');
15539:  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);
15540:  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;
15541:  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);');
15542:  Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)');
15543:  Function ValidGraphic( aGraphic : TGraphic) : Boolean');
15544:  Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor');
15545:  Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor');
15546:  Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor');
15547:  Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor');
15548:  Function MediumColor( Color1, Color2 : TColor) : TColor');
15549:  Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect');
15550:  Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect');
15551:  Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect');
15552:  Procedure InflateRectPercent( var aRect : TRect; withPercent : Double)');
15553:  Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect');
15554:  Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect');
15555:  Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean');
15556:  Function PointInEllispe( const aPt : TPoint; const aRect : TRect) : boolean');
15557:  Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer');
15558: end;
15559:
15560: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15561: begin
15562:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');
15563:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15564:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15565:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15566:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15567:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,'
15568:    +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft)');
15569:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15570:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15571:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
```

```
15572:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15573:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
      bmInvertReverseFromColor);
15574:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
      rjResizeTopLeft,  rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15575:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15576:  cCaptionOrientationWarning','String')('Note that text orientation doesn''t work with all fonts!');
15577: end;
15578:
15579: procedure SIRegister_WinSvc(CL: TPSPascalCompiler);
15580: begin
15581:  Const SERVICES_ACTIVE_DATABASEA','String 'ServicesActive');
15582:  SERVICES_ACTIVE_DATABASEW','String') 'ServicesActive');
15583:  Const SERVICES_ACTIVE_DATABASE','String')' SERVICES_ACTIVE_DATABASEA');
15584:  Const SERVICES_FAILED_DATABASEA','String' 'ServicesFailed');
15585:  Const SERVICES_FAILED_DATABASEW','String' 'ServicesFailed');
15586:  Const SERVICES_FAILED_DATABASE','String' 'SERVICES_FAILED_DATABASEA');
15587:  Const SC_GROUP_IDENTIFIERA','String'). '+');
15588:  Const SC_GROUP_IDENTIFIERW','String') '+');
15589:  Const SC_GROUP_IDENTIFIER','string 'SC_GROUP_IDENTIFIERA');
15590:  Const SERVICE_NO_CHANGE','LongWord $FFFFFFFF);
15591:  Const SERVICE_ACTIVE','LongWord')( $00000001);
15592:  Const SERVICE_INACTIVE','LongWord $00000002);
15593:  Const SERVICE_CONTROL_STOP','LongWord $00000001);
15594:  Const SERVICE_CONTROL_PAUSE','LongWord $00000002);
15595:  Const SERVICE_CONTROL_CONTINUE','LongWord $00000003);
15596:  Const SERVICE_CONTROL_INTERROGATE','LongWord $00000004);
15597:  Const SERVICE_CONTROL_SHUTDOWN','LongWord $00000005);
15598:  Const SERVICE_STOPPED','LongWord $00000001);
15599:  Const SERVICE_START_PENDING','LongWord $00000002);
15600:  Const SERVICE_STOP_PENDING','LongWord $00000003);
15601:  Const SERVICE_RUNNING','LongWord $00000004);
15602:  Const SERVICE_CONTINUE_PENDING','LongWord $00000005);
15603:  Const SERVICE_PAUSE_PENDING','LongWord $00000006);
15604:  Const SERVICE_PAUSED','LongWord $00000007);
15605:  Const SERVICE_ACCEPT_STOP','LongWord $00000001);
15606:  Const SERVICE_ACCEPT_PAUSE_CONTINUE','LongWord $00000002);
15607:  Const SERVICE_ACCEPT_SHUTDOWN','LongWord $00000004);
15608:  Const SC_MANAGER_CONNECT','LongWord $0001);
15609:  Const SC_MANAGER_CREATE_SERVICE','LongWord $0002;
15610:  Const SC_MANAGER_ENUMERATE_SERVICE','LongWord $0004);
15611:  Const SC_MANAGER_LOCK','LongWord $0008);
15612:  Const SC_MANAGER_QUERY_LOCK_STATUS','LongWord $0010);
15613:  Const SC_MANAGER_MODIFY_BOOT_CONFIG','LongWord $0020);
15614:  Const SERVICE_QUERY_CONFIG','LongWord $0001);
15615:  Const SERVICE_CHANGE_CONFIG','LongWord $0002);
15616:  Const SERVICE_QUERY_STATUS','LongWord $0004);
15617:  Const SERVICE_ENUMERATE_DEPENDENTS','LongWord $0008);
15618:  Const SERVICE_START','LongWord $0010);
15619:  Const SERVICE_STOP','LongWord $0020);
15620:  Const SERVICE_PAUSE_CONTINUE','LongWord $0040);
15621:  Const SERVICE_INTERROGATE','LongWord $0080);
15622:  Const SERVICE_USER_DEFINED_CONTROL','LongWord $0100);
15623:  Const SERVICE_KERNEL_DRIVER','LongWord $00000001);
15624:  Const SERVICE_FILE_SYSTEM_DRIVER','LongWord $00000002);
15625:  Const SERVICE_ADAPTER','LongWord $00000004);
15626:  Const SERVICE_RECOGNIZER_DRIVER','LongWord $00000008);
15627:  Const SERVICE_WIN32_OWN_PROCESS','LongWord $00000010);
15628:  Const SERVICE_WIN32_SHARE_PROCESS','LongWord $00000020);
15629:  Const SERVICE_INTERACTIVE_PROCESS','LongWord $00000100);
15630:  Const SERVICE_BOOT_START','LongWord $00000000);
15631:  Const SERVICE_SYSTEM_START','LongWord $00000001);
15632:  Const SERVICE_AUTO_START','LongWord $00000002);
15633:  Const SERVICE_DEMAND_START','LongWord $00000003);
15634:  Const SERVICE_DISABLED','LongWord $00000004);
15635:  Const SERVICE_ERROR_IGNORE','LongWord $00000000);
15636:  Const SERVICE_ERROR_NORMAL','LongWord $00000001);
15637:  Const SERVICE_ERROR_SEVERE','LongWord $00000002);
15638:  Const SERVICE_ERROR_CRITICAL','LongWord $00000003);
15639:   CL.AddTypeS('SC_HANDLE', 'THandle');
15640:   //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15641:   CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15642:   Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15643:    +': DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15644:    +'cificExitCode : DWORD; dwCheckPoint : DWORD; dwWaitHint : DWORD; end');
15645:   Const SERVICE_STATUS', '_SERVICE_STATUS');
15646:   Const TServiceStatus', '_SERVICE_STATUS');
15647:   CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15648:    +'playName : PChar; ServiceStatus : TServiceStatus; end');
15649:   ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15650:   _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15651:   TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15652:   TEnumServiceStatus', 'TEnumServiceStatusA');
15653:   SC_LOCK', '___Pointer');
15654:   _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOner: PChar;dwLockDuration:DWORD;end;
15655:   _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15656:   QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15657:   QUERY_SERVICE_LOCK_STATUS', 'QUERY_SERVICE_LOCK_STATUSA');
15658:   TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
```

```
15659:   //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15660:   TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15661:   _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15662:    +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15663:    +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15664:    +'iceStartName : PChar; lpDisplayName : PChar; end');
15665:   _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15666:   QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15667:   QUERY_SERVICE_CONFIG', 'QUERY_SERVICE_CONFIGA');
15668:   TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15669:   TQueryServiceConfig', 'TQueryServiceConfigA');
15670:  Function CloseServiceHandle( hSCObject : SC_HANDLE) : BOOL');
15671:  Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15672:  Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;'
15673:  +' lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15674:  Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
      dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar; '
15675:  +'lpdwTagId : DWORD; lpDependencies, lpServiceStartName, lpPassword : PChar) : SC_HANDLE');
15676:  Function DeleteService( hService : SC_HANDLE) : BOOL');
15677:  Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
      TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD) : BOOL');
15678:  Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
      lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
      : DWORD):BOOL');
15679:  Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
      lpcchBuffer:DWORD):BOOL');
15680:  Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceNme,lpDisplayName:PChar;var
      lpcchBuffer:DWORD):BOOL;
15681:  Function LockServiceDatabase( hSCManager : SC_HANDLE) : SC_LOCK');
15682:  Function NotifyBootConfigStatus( BootAcceptable : BOOL) : BOOL');
15683:  Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD) : SC_HANDLE');
15684:  Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE');
15685:  Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
      cbBufSize : DWORD; var pcbBytesNeeded : DWORD) : BOOL');
15686:  Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15687:  Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15688:  Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15689:  Function UnlockServiceDatabase( ScLock : SC_LOCK) : BOOL');
15690: end;
15691:
15692: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15693: begin
15694:  Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
      AStartOfWeek : TDayOfWeekName; AWeekends: TDaysOfWeek;AWeekendColor:TColor;
      BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15695:  Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
      DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDat
15696:  Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15697:  Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
      TWinControl;
15698:  Procedure SetupPopupCalendar( PopupCalendar : TWinControl; AStartOfWeek : TDayOfWeekName; AWeekends :
      TDaysOfWeek; AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:
      TDateTime)');
15699:  Function CreateNotifyThread(const
      FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15700: end;
15701:
15702: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15703: begin
15704:   CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15705:   CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15706:  Function NtfsGetCompression2( const FileName : TFileName; var State : Short) : Boolean;');
15707:  Function NtfsGetCompression12( const FileName : TFileName) : TFileCompressionState;');
15708:  Function NtfsSetCompression2( const FileName : TFileName; const State : Short) : Boolean');
15709:  Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)');
15710:  Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)');
15711:  Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)');
15712:  Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)');
15713:  Function NtfsSetSparse2( const FileName : string) : Boolean');
15714:  Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64) : Boolean');
15715:  Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64) : Boolean');
15716:  Function NtfsSparseStreamsSupported2( const Volume : string) : Boolean');
15717:  Function NtfsGetSparse2( const FileName : string) : Boolean');
15718:  Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD) : Boolean');
15719:  Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword) : Boolean');
15720:  Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD) : Boolean');
15721:  Function NtfsReparsePointsSupported2( const Volume : string) : Boolean');
15722:  Function NtfsFileHasReparsePoint2( const Path : string) : Boolean');
15723:  Function NtfsIsFolderMountPoint2( const Path : string) : Boolean');
15724:  Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char) : Boolean');
15725:  Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString) : Boolean');
15726:   CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15727:  Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15728:  Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15729:  Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15730:  Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
15731:  Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped):Boolean');
15732:  Function NtfsCreateJunctionPoint2( const Source, Destination : string) : Boolean');
```

```
15733:  Function NtfsDeleteJunctionPoint2( const Source : string) : Boolean');
15734:  Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15735:   CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15736:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15737:   CL.AddTypeS('TStreamIds', 'set of TStreamId');
15738:   TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15739:   CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15740:   +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15741:  Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15742:  Function NtfsFindNextStream2( var Data : TFindStreamData) : Boolean');
15743:  Function NtfsFindStreamClose2( var Data : TFindStreamData) : Boolean');
15744:  Function NtfsCreateHardLink2( const LinkFileName, ExistingFileName : String) : Boolean');
15745:  Function NtfsCreateHardLinkA2( const LinkFileName, ExistingFileName : AnsiString) : Boolean');
15746:  Function NtfsCreateHardLinkW2( const LinkFileName, ExistingFileName : WideString) : Boolean');
15747:   CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15748:  Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
15749:  Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
      List:TStrings):Bool
15750:  Function NtfsDeleteHardLinks2( const FileName : string) : Boolean');
15751:   FindClass('TOBJECT'),'EJclFileSummaryError');
15752:   TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )');
15753:   TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )');
15754:   TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID) : Boolean');
15755:   CL.AddClassN(CL.FindClass('TOBJECT'),'TJclFileSummary');
15756:   SIRegister_TJclFilePropertySet(CL);
15757:   //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15758:   SIRegister_TJclFileSummary(CL);
15759:   SIRegister_TJclFileSummaryInformation(CL);
15760:   SIRegister_TJclDocSummaryInformation(CL);
15761:   SIRegister_TJclMediaFileSummaryInformation(CL);
15762:   SIRegister_TJclMSISummaryInformation(CL);
15763:   SIRegister_TJclShellSummaryInformation(CL);
15764:   SIRegister_TJclStorageSummaryInformation(CL);
15765:   SIRegister_TJclImageSummaryInformation(CL);
15766:   SIRegister_TJclDisplacedSummaryInformation(CL);
15767:   SIRegister_TJclBriefCaseSummaryInformation(CL);
15768:   SIRegister_TJclMiscSummaryInformation(CL);
15769:   SIRegister_TJclWebViewSummaryInformation(CL);
15770:   SIRegister_TJclMusicSummaryInformation(CL);
15771:   SIRegister_TJclDRMSummaryInformation(CL);
15772:   SIRegister_TJclVideoSummaryInformation(CL);
15773:   SIRegister_TJclAudioSummaryInformation(CL);
15774:   SIRegister_TJclControlPanelSummaryInformation(CL);
15775:   SIRegister_TJclVolumeSummaryInformation(CL);
15776:   SIRegister_TJclShareSummaryInformation(CL);
15777:   SIRegister_TJclLinkSummaryInformation(CL);
15778:   SIRegister_TJclQuerySummaryInformation(CL);
15779:   SIRegister_TJclImageInformation(CL);
15780:   SIRegister_TJclJpegSummaryInformation(CL);
15781: end;
15782:
15783: procedure SIRegister_Jcl8087(CL: TPSPascalCompiler);
15784: begin
15785:   AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15786:   T8087Rounding','( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate )');
15787:   T8087Infinity','( icProjective, icAffine )');
15788:   T8087Exception','(emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15789:   CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15790:  Function Get8087ControlWord : Word');
15791:  Function Get8087Infinity : T8087Infinity');
15792:  Function Get8087Precision : T8087Precision');
15793:  Function Get8087Rounding : T8087Rounding');
15794:  Function Get8087StatusWord( ClearExceptions : Boolean) : Word');
15795:  Function Set8087Infinity( const Infinity : T8087Infinity) : T8087Infinity');
15796:  Function Set8087Precision( const Precision : T8087Precision) : T8087Precision');
15797:  Function Set8087Rounding( const Rounding : T8087Rounding) : T8087Rounding');
15798:  Function Set8087ControlWord( const Control : Word) : Word');
15799:  Function ClearPending8087Exceptions : T8087Exceptions');
15800:  Function GetPending8087Exceptions : T8087Exceptions');
15801:  Function GetMasked8087Exceptions : T8087Exceptions');
15802:  Function SetMasked8087Exceptions(Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15803:  Function Mask8087Exceptions( Exceptions:T8087Exceptions) : T8087Exceptions');
15804:  Function Unmask8087Exceptions(Exceptions:T8087Exceptions;ClearBefore : Boolean) : T8087Exceptions');
15805: end;
15806:
15807: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15808: begin
15809:  CL.AddDelphiFunction('Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)');
15810:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)');
15811:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
15812:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)');
15813:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)');
15814:  Procedure BoxSetItem( List : TWinControl; Index : Integer)');
15815:  Function BoxGetFirstSelection( List : TWinControl) : Integer');
15816:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean');
15817: end;
15818:
15819: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
```

```
15820: begin
15821:  //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context');
15822:  //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee');
15823:  CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6);
15824:   type ULONG', 'Cardinal');
15825:      LPCWSTR', 'PChar');
15826:   CL.AddTypeS('LPWSTR', 'PChar');
15827:   LPSTR', 'PChar');
15828:   TBindVerb', 'ULONG');
15829:   TBindInfoF', 'ULONG');
15830:   TBindF', 'ULONG');
15831:   TBSCF', 'ULONG');
15832:   TBindStatus', 'ULONG');
15833:   TCIPStatus', 'ULONG');
15834:   TBindString', 'ULONG');
15835:   TPiFlags', 'ULONG');
15836:   TOIBdgFlags', 'ULONG');
15837:   TParseAction', 'ULONG');
15838:   TPSUAction', 'ULONG');
15839:   TQueryOption', 'ULONG');
15840:   TPUAF', 'ULONG');
15841:   TSZMFlags', 'ULONG');
15842:   TUrlZone', 'ULONG');
15843:   TUrlTemplate', 'ULONG');
15844:   TZAFlags', 'ULONG');
15845:   TUrlZoneReg', 'ULONG');
15846:  'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001);
15847:  CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002);
15848:  const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004);
15849:  const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008);
15850:  const 'CF_NULL','LongInt').SetInt( 0);
15851:  const 'CFSTR_MIME_NULL','LongInt').SetInt( 0);
15852:  const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain');
15853:  const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext');
15854:  const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-xbitmap');
15855:  const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript');
15856:  const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff');
15857:  const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic');
15858:  const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav');
15859:  const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav');
15860:  const 'CFSTR_MIME_GIF','String').SetString( 'image/gif');
15861:  const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg');
15862:  const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg');
15863:  const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff');
15864:  const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png');
15865:  const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp');
15866:  const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg');
15867:  const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf');
15868:  const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf');
15869:  const 'CFSTR_MIME_AVI','String').SetString( 'video/avi');
15870:  const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg');
15871:  const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals');
15872:  const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream');
15873:  const 'CFSTR_MIME_RAWDATASTRM','String').SetString( 'application/octet-stream');
15874:  const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf');
15875:  const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff');
15876:  const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio');
15877:  const 'CFSTR_MIME_XBM','String').SetString( 'image/xbm');
15878:  const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime');
15879:  const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo');
15880:  const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString( 'video/x-sgi-movie');
15881:  const 'CFSTR_MIME_HTML','String').SetString( 'text/html');
15882:  const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15883:  const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15884:  const 'E_PENDING','LongWord').SetUInt( $8000000A);
15885:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15886:   SIRegister_IPersistMoniker(CL);
15887:   SIRegister_IBindProtocol(CL);
15888:   SIRegister_IBinding(CL);
15889:  const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15890:  const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15891:  const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15892:  const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15893:  const 'BINDINFOF_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15894:  const 'BINDINFOF_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15895:  const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15896:  const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15897:  const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
15898:  const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15899:  const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15900:  const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15901:  const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
15902:  const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
15903:  const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
15904:  const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
15905:  const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
15906:  const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
15907:  const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
15908:  const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
```

```
15909:  const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
15910:  const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
15911:  const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
15912:  const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
15913:  //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
15914:  //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
15915:  //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
15916:  const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
15917:  const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
15918:  const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
15919:  const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
15920:  const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
15921:  const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
15922:  const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
15923:  const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
15924:  const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
15925:  const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
15926:  const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
15927:  const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
15928:  const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
15929:  const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
15930:  const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
15931:  const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
15932:  const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
15933:  const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
15934:  const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
15935:  const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
15936:  const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
15937:  const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
15938:  const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
15939:  const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
15940:  const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
15941:  const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
15942:  const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
15943:  const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
15944:  const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
15945:  const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
15946:  const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
15947:  const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
15948:  const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
15949:  const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
15950:  const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
15951:  const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
15952:  const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
15953:  const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
15954:  const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
15955:  const 'BINDSTATUS_COMPACT_POLICY_RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
15956:  const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY_RECEIVED + 1);
15957:  const 'BINDSTATUS_COOKIE_STATE_UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
15958:  const 'BINDSTATUS_COOKIE_STATE_ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_UNKNOWN + 1);
15959:  const 'BINDSTATUS_COOKIE_STATE_REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_ACCEPT + 1);
15960:  const 'BINDSTATUS_COOKIE_STATE_PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_REJECT + 1);
15961:  const 'BINDSTATUS_COOKIE_STATE_LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_PROMPT + 1);
15962:  const 'BINDSTATUS_COOKIE_STATE_DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_LEASH + 1);
15963:  const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE_DOWNGRADE + 1);
15964:  const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
15965:  const 'BINDSTATUS_SESSION_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
15966:  const 'BINDSTATUS_PERSISTENT_COOKIE_RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE_RECEIVED + 1);
15967:  const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE_RECEIVED + 1);
15968:  const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);
15969:  const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
15970:  const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
15971:  const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
15972:  const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
15973:  // PBindInfo', '^TBindInfo // will not work');
15974:   {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
15975:    +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
15976:    +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
15977:    +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
15978:    +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
15979:   TBindInfo', '_tagBINDINFO');
15980:   BINDINFO', '_tagBINDINFO');}
15981:   _REMSECURITY_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
15982:    +'criptor : DWORD; bInheritHandle : BOOL; end');
15983:   TRemSecurityAttributes', '_REMSECURITY_ATTRIBUTES');
15984:   REMSECURITY_ATTRIBUTES', '_REMSECURITY_ATTRIBUTES');
15985:   //PRemBindInfo', '^TRemBindInfo // will not work');
15986:   {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
15987:    +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
15988:    +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
15989:    +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknow'
15990:    +'n; dwReserved : DWORD; end');
15991:   TRemBindInfo', '_tagRemBINDINFO');
15992:   RemBINDINFO', '_tagRemBINDINFO');}
15993:   //PRemFormatEtc', '^TRemFormatEtc // will not work');
15994:   tagRemFORMATETC','record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tymed:DWORD; end');
15995:   TRemFormatEtc', 'tagRemFORMATETC');
15996:   RemFORMATETC', 'tagRemFORMATETC');
15997:   SIRegister_IBindStatusCallback(CL);
```

```
15998:   SIRegister_IAuthenticate(CL);
15999:   SIRegister_IHttpNegotiate(CL);
16000:   SIRegister_IWindowForBindingUI(CL);
16001:  const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16002:  const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16003:  const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16004:  const 'CIP_OLDER_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16005:  const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_OLDER_VERSION_EXISTS + 1);
16006:  const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16007:  const 'CIP_EXE_SELF_REGISTERATION_TIMEOUT',LongInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16008:  const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTERATION_TIMEOUT + 1);
16009:  const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16010:  const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16011:   SIRegister_ICodeInstall(CL);
16012:   SIRegister_IWinInetInfo(CL);
16013:  const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16014:   SIRegister_IHttpSecurity(CL);
16015:   SIRegister_IWinInetHttpInfo(CL);
16016:   SIRegister_IBindHost(CL);
16017:  const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16018:  const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16019:  const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16020:  Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback) : HResult');
16021:  Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback) : HResult');
16022:  Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
       : IBindStatusCallback) : HResult');
16023:  Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
       IBindStatusCallback) : HResult');
16024:  Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
       p3:IStream;p4:DWORD;p5:IBindStatusCallback):HResult');
16025:  Function HlinkGoBack( unk : IUnknown) : HResult');
16026:  Function HlinkGoForward( unk : IUnknown) : HResult');
16027:  Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR) : HResult');
16028: // Function HlinkNavigateMoniker( Unk : IUnknown; mkTarget : IMoniker) : HResult');
16029:   SIRegister_IInternet(CL);
16030:  const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16031:  const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16032:  const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16033:  const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16034:  const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16035:  const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16036:  const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16037:  const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16038:  const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16039:  const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16040:  const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16041:  const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16042:  const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16043:  const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16044:   //POLEStrArray', '^TOLESTRArray // will not work');
16045:   SIRegister_IInternetBindInfo(CL);
16046:  const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16047:  const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16048:  const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16049:  const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16050:  const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16051:  const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16052:  const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16053:  const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16054:  const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16055:  const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16056:  const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16057:  //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16058:   //PProtocolData', '^TProtocolData // will not work');
16059:   _tagPROTOCOLDATA','record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16060:   TProtocolData', '_tagPROTOCOLDATA');
16061:   PROTOCOLDATA', '_tagPROTOCOLDATA');
16062:   CL.AddInterface(CL.FindInterface('IUNKNOWN'),IInternetProtocolSink, 'IInternetProtocolSink');
16063:   SIRegister_IInternetProtocolRoot(CL);
16064:   SIRegister_IInternetProtocol(CL);
16065:   SIRegister_IInternetProtocolSink(CL);
16066:  const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16067:   SIRegister_IInternetSession(CL);
16068:   SIRegister_IInternetThreadSwitch(CL);
16069:   SIRegister_IInternetPriority(CL);
16070:  const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16071:  const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16072:  const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16073:  const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16074:  const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16075:  const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16076:  const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16077:  const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16078:  const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16079:  const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16080:  const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16081:  const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16082:  const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16083:  const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
```

```
16084:  const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16085:  const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16086:  const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16087:  const 'PSU_DEFAULT','LongInt').SetInt( 1);
16088:  const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16089:  const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16090:  const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16091:  const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16092:  const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16093:  const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16094:  const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16095:  const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16096:  const 'QUERY_USES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16097:  const 'QUERY_IS_CACHED','LongInt').SetInt( QUERY_USES_NETWORK + 1);
16098:  const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16099:  const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16100:  const 'QUERY_USES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16101:  const 'QUERY_IS_SECURE','LongInt').SetInt( QUERY_USES_CACHE + 1);
16102:  const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16103:   SIRegister_IInternetProtocolInfo(CL);
16104:   IOInet', 'IInternet');
16105:   IOInetBindInfo', 'IInternetBindInfo');
16106:   IOInetProtocolRoot', 'IInternetProtocolRoot');
16107:   IOInetProtocol', 'IInternetProtocol');
16108:   IOInetProtocolSink', 'IInternetProtocolSink');
16109:   IOInetProtocolInfo', 'IInternetProtocolInfo');
16110:   IOInetSession', 'IInternetSession');
16111:   IOInetPriority', 'IInternetPriority');
16112:   IOInetThreadSwitch', 'IInternetThreadSwitch');
16113:  Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16114:  Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16115:  Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult');
16116:  Function CoInternetGetProtocolFlags(pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16117:  Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD;
       pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16118:  Function CoInternetGetSession(dwSessionMode:DWORD; var pIInternetSession:
       IInternetSes;dwReserved:DWORD):HResult;
16119:  Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var
       pwzSecUrl:LPWSTR;psuAct:TPSUAction;dwReserv:DWORD):HResult;
16120:  Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16121:  Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult :
       LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16122:  Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult');
16123:  Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16124:  Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer :
       TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16125:  Function OInetGetSession(dwSessionMode:DWORD; var
       pIInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16126:   //Function CopyBindInfo( const cbiSrc : TBindInfo; var biDest : TBindInfo) : HResult');
16127:  //Procedure ReleaseBindInfo( const bindinfo : TBindInfo)');
16128: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16129: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16130:  //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16131:  //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16132:  //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16133:  const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16134:   SIRegister_IInternetSecurityMgrSite(CL);
16135:  const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16136:  const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16137:  const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16138:  const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16139:  const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16140:  const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16141:  const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16142:  const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16143:  const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16144:  const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16145:  const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16146:  const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16147:  const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16148:  const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16149:  const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16150:  const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16151:  const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16152:  const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16153:  const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16154:  const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16155:  const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16156:  const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16157:  const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16158:  const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16159:  const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16160:  const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16161:  const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16162:   SIRegister_IInternetSecurityManager(CL);
16163:   SIRegister_IInternetHostSecurityManager(CL);
```

```
16164:   SIRegister_IInternetSecurityManagerEx(CL);
16165:  const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16166:  const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16167:  const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16168:  const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16169:  const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16170:  const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16171:  const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16172:  const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16173:  const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16174:  const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16175:  const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16176:  const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16177:  const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16178:  const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16179:  const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16180:  const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16181:  const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16182:  const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16183:  const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16184:  const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16185:  const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16186:  const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16187:  const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16188:  const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16189:  const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16190:  const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16191:  const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16192:  const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16193:  const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16194:  const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16195:  const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16196:  const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16197:  const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16198:  const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16199:  const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16200:  const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16201:  const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16202:  const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16203:  const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16204:  const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16205:  const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16206:  const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16207:  const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16208:  const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019ff);
16209:  const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16210:  const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16211:  const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16212:  const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16213:  const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16214:  const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16215:  const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16216:  const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16217:  const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16218:  const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16219:  const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16220:  const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16221:  const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16222:  const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16223:  const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);
16224:  const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16225:  const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16226:  const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16227:  const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16228:  const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16229:  const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16230:  const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16231:  const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16232:  const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16233:  const 'URLACTION_INFODELIVERY_NO_REMOVING_CHANNELS','LongWord').SetUInt( $00001D02);
16234:  const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16235:  const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16236:  const 'URLACTION_INFODELIVERY_NO_REMOVING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16237:  const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16238:  const 'URLACTION_INFODELIVERY_CURR_MAX','LongWord').SetUInt( $00001D06);
16239:  const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16240:  const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16241:  const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16242:  const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16243:  const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16244:  const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16245:  const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16246:  const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16247:  const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16248:  const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16249:  const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16250:  const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16251:  const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16252:  const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
```

```
16253:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16254:  const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16255:  const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16256:  const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16257:  const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16258:  const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16259:  const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16260:  const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16261:  const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16262:  const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16263:  const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16264:  const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16265:  Function GetUrlPolicyPermissions( dw : DWORD) : DWORD');
16266:  Function SetUrlPolicyPermissions( dw, dw2 : DWORD) : DWORD');
16267:  const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16268:  const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16269:  const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16270:  const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16271:  const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16272:  const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16273:  const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16274:  const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16275:  const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16276:  const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16277:  const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16278:  const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16279:  const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16280:  const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16281:  const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16282:  const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16283:  const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16284:  const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16285:  const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16286:  const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16287:  const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16288:  const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16289:  const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16290:  const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16291:  const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16292:  const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16293:   //PZoneAttributes', '^TZoneAttributes // will not work');
16294:  _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');
16295:   { _ZONEATTRIBUTES = packed record
16296:     cbSize: ULONG;
16297:     szDisplayName: array [0..260 - 1] of WideChar;
16298:     szDescription: array [0..200 - 1] of WideChar;
16299:     szIconPath: array [0..260 - 1] of WideChar;
16300:     dwTemplateMinLevel: DWORD;
16301:     dwTemplateRecommended: DWORD;
16302:     dwTemplateCurrentLevel: DWORD;
16303:     dwFlags: DWORD;
16304:   end; }
16305:   TZoneAttributes', '_ZONEATTRIBUTES');
16306:   ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16307:  const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16308:  const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16309:  const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16310:   SIRegister_IInternetZoneManager(CL);
16311:   SIRegister_IInternetZoneManagerEx(CL);
16312:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16313:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16314:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16315:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16316:  const 'SOFTDIST_ADSTATE_NONE','LongWord').SetUInt( $00000000);
16317:  const 'SOFTDIST_ADSTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16318:  const 'SOFTDIST_ADSTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16319:  const 'SOFTDIST_ADSTATE_INSTALLED','LongWord').SetUInt( $00000003);
16320:   //PCodeBaseHold', '^TCodeBaseHold // will not work');
16321:   _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR; '
16322:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16323:   TCodeBaseHold', '_tagCODEBASEHOLD');
16324:   CODEBASEHOLD', '_tagCODEBASEHOLD');
16325:   //PSoftDistInfo', '^TSoftDistInfo // will not work');
16326:   _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAd'
16327:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI'
16328:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
16329:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdve'
16330:    +'rtisedVersionLS : DWORD; dwReserved : DWORD; end');
16331:   TSoftDistInfo', '_tagSOFTDISTINFO');
16332:   SOFTDISTINFO', '_tagSOFTDISTINFO');
16333:   SIRegister_ISoftDistExt(CL);
16334:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo) : HResult');
16335:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
       dwAdvertisedVersionLS : DWORD) : HResult');
16336:   SIRegister_IDataFilter(CL);
16337:  // PProtocolFilterData', '^TProtocolFilterData // will not work');
16338:    _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink : '
```

```
16339:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil'
16340:    +'terFlags : DWORD; end');
16341:   TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16342:   PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16343:   //PDataInfo', '^TDataInfo // will not work');
16344:   _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL'
16345:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16346:   TDataInfo', '_tagDATAINFO');
16347:   DATAINFO', '_tagDATAINFO');
16348:   SIRegister_IEncodingFilterFactory(CL);
16349:  Function IsLoggingEnabled( pszUrl : PChar) : BOOL');
16350:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar) : BOOL');
16351:  //Function IsLoggingEnabledW( pszUrl : PWideChar) : BOOL');
16352:   //PHitLoggingInfo', '^THitLoggingInfo // will not work');
16353:   _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU'
16354:    +rlName : LPSTR; StartTime : TSystemTime;EndTime:TSystemTime; lpszExtendedInfo : LPSTR; end');
16355:   THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16356:   HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16357:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo) : BOOL');
16358: end;
16359:
16360: procedure SIRegister_DFFUtils(CL: TPSPascalCompiler);
16361: begin
16362:  Procedure reformatMemo( const m : TCustomMemo)');
16363:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer)');
16364:  Procedure MoveToTop( memo : TMemo)');
16365:  Procedure ScrollToTop( memo : TMemo)');
16366:  Function LineNumberClicked( memo : TMemo) : integer');
16367:  Function MemoClickedLine( memo : TMemo) : integer');
16368:  Function ClickedMemoLine( memo : TMemo) : integer');
16369:  Function MemoLineClicked( memo : TMemo) : integer');
16370:  Function LinePositionClicked( Memo : TMemo) : integer');
16371:  Function ClickedMemoPosition( memo : TMemo) : integer');
16372:  Function MemoPositionClicked( memo : TMemo) : integer');
16373:  Procedure AdjustGridSize( grid : TDrawGrid)');
16374:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer)');
16375:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer)');
16376:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer);');
16377:  Procedure Sortgrid1(Grid:TStringGrid; const SortCol : integer; sortascending : boolean);');
16378:  Procedure sortstrDown( var s : string)');
16379:  Procedure sortstrUp( var s : string)');
16380:  Procedure rotatestrleft( var s : string)');
16381:  Function dffstrtofloatdef( s : string; default : extended) : extended');
16382:  Function deblank( s : string) : string');
16383:  Function IntToBinaryString( const n : integer; MinLength : integer) : string');
16384:  Procedure FreeAndClearListBox( C : TListBox);');
16385:  Procedure FreeAndClearMemo( C : TMemo);');
16386:  Procedure FreeAndClearStringList( C : TStringList);');
16387:  Function dffgetfilesize( f : TSearchrec) : int64');
16388: end;
16389:
16390: procedure SIRegister_MathsLib(CL: TPSPascalCompiler);
16391: begin
16392:   CL.AddTypeS('intset', 'set of byte');
16393:   TPoint64', 'record x : int64; y : int64; end');
16394:  Function GetNextPandigital( size : integer; var Digits : array of integer) : boolean');
16395:  Function IsPolygonal( T : int64; var rank : array of integer) : boolean');
16396:  Function GeneratePentagon( n : integer) : integer');
16397:  Function IsPentagon( p : integer) : boolean');
16398:  Function isSquare( const N : int64) : boolean');
16399:  Function isCube( const N : int64) : boolean');
16400:  Function isPalindrome( const n : int64) : boolean;');
16401:  Function isPalindrome1( const n : int64; var len : integer) : boolean;');
16402:  Function GetEulerPhi( n : int64) : int64');
16403:  Function dffIntPower( a, b : int64) : int64;');
16404:  Function IntPower1( a : extended; b : int64) : extended;');
16405:  Function gcd2( a, b : int64) : int64');
16406:  Function GCDMany( A : array of integer) : integer');
16407:  Function LCMMany( A : array of integer) : integer');
16408:  Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16409:  Function dffFactorial( n : int64) : int64');
16410:  Function digitcount( n : int64) : integer');
16411:  Function nextpermute( var a : array of integer) : boolean');
16412:  Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16413:  Function convertStringToDecimal( s : string; var n : extended) : Boolean');
16414:  Function InttoBinaryStr( nn : integer) : string');
16415:  Function StrtoAngle( const s : string; var angle : extended) : boolean');
16416:  Function AngleToStr( angle : extended) : string');
16417:  Function deg2rad( deg : extended) : extended');
16418:  Function rad2deg( rad : extended) : extended');
16419:  Function GetLongToMercProjection( const long : extended) : extended');
16420:  Function GetLatToMercProjection( const Lat : Extended) : Extended');
16421:  Function GetMercProjectionToLong( const ProjLong : extended) : extended');
16422:  Function GetMercProjectionToLat( const ProjLat : extended) : extended');
16423:   SIRegister_TPrimes(CL);
16424:   //RIRegister_TPrimes(CL);
16425:  //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16426:  CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16427:  Function Random64( const N : Int64) : Int64;');
```

```
16428:  Procedure Randomize64');
16429:  Function Random641 : extended;');
16430:  Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)');
16431: end;
16432:
16433: procedure SIRegister_UGeometry(CL: TPSPascalCompiler);
16434: begin
16435:   TrealPoint', 'record x : extended; y : extended; end');
16436:   Tline', 'record p1 : TPoint; p2 : TPoint; end');
16437:   TRealLine', 'record p1 : TRealPoint; p2 : TRealPoint; end');
16438:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16439:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16440:   PPResult', '( PPoutside, PPInside, PPVertex, PPEdge, PPError )');
16441:  Function realpoint( x, y : extended) : TRealPoint');
16442:  Function dist( const p1, p2 : TrealPoint) : extended');
16443:  Function intdist( const p1, p2 : TPoint) : integer');
16444:  Function dffLine( const p1, p2 : TPoint) : Tline;');
16445:  Function Line1( const p1, p2 : TRealPoint) : TRealline;');
16446:  Function dffCircle( const cx, cy, R : integer) : TCircle;');
16447:  Function Circle1( const cx, cy, R : extended) : TRealCircle;');
16448:  Function GetTheta( const L : TLine) : extended;');
16449:  Function GetTheta1( const p1, p2 : TPoint) : extended;');
16450:  Function GetTheta2( const p1, p2 : TRealPoint) : extended;');
16451:  Procedure Extendline( var L : TLine; dist : integer);');
16452:  Procedure Extendline1( var L : TRealLine; dist : extended);');
16453:  Function Linesintersect( line1, line2 : TLine) : boolean');
16454:  Function ExtendedLinesIntersect(Line1,Line2:TLine; const extendlines:bool;var IP:TPoint):bool;
16455:  Function ExtendedLinesIntersect1(const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint):bool;
16456:  Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint) : boolean');
16457:  Function PointPerpendicularLine( L : TLine; P : TPoint) : TLine');
16458:  Function PerpDistance( L : TLine; P : TPoint) : Integer');
16459:  Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended) : TLine;');
16460:  Function
       AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16461:  Function PointInPoly( const p : TPoint; Points : array of TPoint) : PPResult');
16462:  Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
       Clockwise:bool):integer;
16463:  Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
       const screenCoordinates : boolean; const inflateby : integer)');
16464:  Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16465:  Function DegtoRad( d : extended) : extended');
16466:  Function RadtoDeg( r : extended) : extended');
16467:  Procedure TranslateLeftTo( var L : TLine; newend : TPoint);');
16468:  Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint);');
16469:  Procedure RotateRightEndBy( var L : TLine; alpha : extended)');
16470:  Procedure RotateRightEndTo( var L : TLine; alpha : extended);');
16471:  Procedure RotateRightEndTo1( var p1, p2 : Trealpoint; alpha : extended);');
16472:  Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint) : boolean;');
16473:  Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint) : boolean;');
16474:  Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine) : boolean');
16475:  Function CircleCircleExtTangentLines(C1,C2:TCircle;var C3:TCircle;var L1,L2,PL1,PL2,TL1,Tl2:TLine):Bool;
16476: end;
16477:
16478:
16479: procedure SIRegister_UAstronomy(CL: TPSPascalCompiler);
16480: begin
16481:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGalLonLat )');
16482:   TDTType', '( ttLocal, ttUT, ttGST, ttLST )');
16483:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16484:   TSunrec', 'record TrueEclLon:extended;
       AppEclLon:extended;AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
       TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end;
16485:   TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16486:    +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16487:    +'arth : extended; Phase : extended; end');
16488:   TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16489:    +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDateTime; end');
16490:   TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16491:    +'ct : TDatetime; LastContact : TDateTime; Magnitude:Extended;MaxeclipseUTime:TDateTime;end');
16492:   TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16493:   TPlanetRec', 'record AsOf : TDateTime; Name : string; MeanLon : '
16494:    +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentrici'
16495:    +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMa'
16496:    +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16497:   TPlanetLocRec', 'record PlanetBaseData : TPlanetrec; HelioCentri'
16498:    +'cLonLat : TRpoint; RadiusVector : extended; UncorrectedEarthDistance : ext'
16499:    +'ended; GeoEclLonLat :TRpoint; CorrectedEarthDistance:extended; ApparentRaDecl:TRPoint; end');
16500:   SIRegister_TAstronomy(CL);
16501:  Function AngleToStr( angle : extended) : string');
16502:  Function StrToAngle( s : string; var angle : extended) : boolean');
16503:  Function HoursToStr24( t : extended) : string');
16504:  Function RPoint( x, y : extended) : TRPoint');
16505:  Function getStimename( t : TDTType) : string');
16506: end;
16507:
16508: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16509: begin
16510:   TCardValue', 'Integer');
16511:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
```

```
16512:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16513:   Type(TDecks,Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16514:   SIRegister_TCard(CL);
16515:   SIRegister_TDeck(CL);
16516: end;
16517:
16518: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16519: begin
16520:   tMethodCall', 'Procedure');
16521:   tVerboseCall', 'Procedure ( s : string)');
16522:  // PTEdge', '^TEdge // will not work');
16523:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16524:    +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16525:   SIRegister_TNode(CL);
16526:   SIRegister_TGraphList(CL);
16527: end;
16528:
16529: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16530: begin
16531:   ParserFloat', 'extended');
16532:   //PParserFloat', '^ParserFloat // will not work');
16533:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16534:    +'ulo, IntDiv, IntDIVZ, integerpower, realpower,square,third,fourth,FuncOneVar,FuncTwoVar )');
16535:   //POperation', '^TOperation // will not work');
16536:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16537:    +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16538:    +'; Token : TDFFToken; end');
16539:   TMathProcedure','procedure(AnOperation: TDFFOperation)');
16540:   (CL.FindClass('TOBJECT'),'EMathParserError');
16541:   CL.FindClass('TOBJECT'),'ESyntaxError');
16542:   (CL.FindClass('TOBJECT'),'EExpressionHasBlanks');
16543:   (CL.FindClass('TOBJECT'),'EExpressionTooComplex');
16544:   (CL.FindClass('TOBJECT'),'ETooManyNestings');
16545:   (CL.FindClass('TOBJECT'),'EMissMatchingBracket');
16546:   (CL.FindClass('TOBJECT'),'EBadName');
16547:   (CL.FindClass('TOBJECT'),'EParserInternalError');
16548:   ('TExParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16549:   SIRegister_TCustomParser(CL);
16550:   SIRegister_TExParser(CL);
16551: end;
16552:
16553:   function isService: boolean;
16554:   begin
16555:     result:= NOT(Application is TApplication);
16556:     {result:= Application is TServiceApplication;}
16557:   end;
16558:   function isApplication: boolean;
16559:   begin
16560:     result:= Application is TApplication;
16561:   end;
16562:   //SM_REMOTESESSION = $1000
16563:   function isTerminalSession: boolean;
16564:   begin
16565:     result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16566:   end;
16567:
16568: {A simple Oscilloscope using  TWaveIn class.
16569:  More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
16570: uses
16571:   Forms,
16572:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},
16573:   ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
16574:   uColorFunctions in 'uColorFunctions.pas',
16575:   AMixer in 'AMixer.pas',
16576:   uSettings in 'uSettings.pas',
16577:   UWavein4 in 'UWavein4.pas',
16578:   U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
16579:   ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
16580:
16581:
16582: Functions_max hex in the box maXbox
16583: functionslist.txt
16584: FunctionsList1 3.9.9.86/88/91/92/94/95
16585:
16586: ********************************************************************************
16587: Procedure
16588: PROCEDURE SIZE 7507 7401 6792 6310 5971 4438 3797 3600
16589: Procedure ***************************Now the Procedure list******************
16590: Procedure ( ACol, ARow : Integer; Items : TStrings)
16591: Procedure ( Agg : TAggregate)
16592: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
16593: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
16594: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
16595: Procedure ( ASender : TObject; const ABytes : Integer)
16596: Procedure ( ASender : TObject; VStream : TStream)
16597: Procedure ( AThread : TIdThread)
16598: Procedure ( AWebModule : TComponent)
16599: Procedure ( Column : TColumn)
16600: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
```

```
16601: Procedure ( const iStart : integer; const sText : string)
16602: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
16603: Procedure ( Database : TDatabase; LoginParams : TStrings)
16604: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var
       Action:TReconcileAction)
16605: Procedure ( DATASET : TDATASET)
16606: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
16607: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
16608: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
16609: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
16610: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
16611: Procedure ( Done : Integer)
16612: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
16613: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
16614: Procedure
       (HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
16615: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
16616: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
16617: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
16618: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
16619: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
16620: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
16621: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
16622: Procedure ( SENDER : TFIELD; const TEXT : String)
16623: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
16624: Procedure ( Sender : TIdTelnet; const Buffer : String)
16625: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
16626: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
16627: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
16628: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
16629: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
16630: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
16631: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
16632: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
16633: Procedure ( Sender : TObject; Button : TMPBtnType)
16634: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
16635: Procedure ( Sender : TObject; Button : TUDBtnType)
16636: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
16637: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
16638: Procedure ( Sender : TObject; Column : TListColumn)
16639: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
16640: Procedure ( Sender : TObject; Connecting : Boolean)
16641: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
       DoneDraw:Bool
16642: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
16643: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
16644: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
16645: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
16646: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
16647: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
16648: Procedure ( Sender : TObject; Index : LongInt)
16649: Procedure ( Sender : TObject; Item : TListItem)
16650: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
16651: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
16652: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
16653: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
16654: Procedure ( Sender : TObject; Item : TListItem; var S : string)
16655: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
16656: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
16657: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
16658: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
16659: Procedure ( Sender : TObject; Node : TTreeNode)
16660: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
16661: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
16662: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
16663: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
16664: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
16665: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
16666: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
16667: Procedure ( Sender : TObject; Rect : TRect)
16668: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
16669: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
16670: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
16671: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
16672: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
16673: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
16674: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
16675: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
16676: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
16677: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
16678: Procedure ( Sender : TObject; Thread : TServerClientThread)
16679: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
16680: Procedure ( Sender : TObject; Username, Password : string)
16681: Procedure ( Sender : TObject; var AllowChange : Boolean)
16682: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
16683: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
16684: Procedure ( Sender : TObject; var Continue : Boolean)
16685: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
16686: Procedure ( Sender : TObject; var Username : string)
```

```
16687: Procedure ( Sender : TObject; Wnd : HWND)
16688: Procedure ( Sender : TToolbar; Button : TToolButton)
16689: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
16690: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
16691: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
16692: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
16693: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
16694: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
16695: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
16696: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
16697: procedure (Sender: TObject)
16698: procedure (Sender: TObject; var Done: Boolean)
16699: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
16700: procedure _T(Name: tbtString; v: Variant);
16701: Procedure AbandonSignalHandler( RtlSigNum : Integer)
16702: Procedure Abort
16703: Procedure About1Click( Sender : TObject)
16704: Procedure Accept( Socket : TSocket)
16705: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
16706: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
16707: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
16708: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
16709: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
16710: Procedure Add( Addend1, Addend2 : TMyBigInt)
16711: Procedure ADD( const AKEY, AVALUE : VARIANT)
16712: Procedure Add( const Key : string; Value : Integer)
16713: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
16714: Procedure ADD( FIELD : TFIELD)
16715: Procedure ADD( ITEM : TMENUITEM)
16716: Procedure ADD( POPUP : TPOPUPMENU)
16717: Procedure AddCharacters( xCharacters : TCharSet)
16718: Procedure AddDriver( const Name : string; List : TStrings)
16719: Procedure AddImages( Value : TCustomImageList)
16720: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
16721: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
16722: Procedure AddLoader( Loader : TBitmapLoader)
16723: Procedure ADDPARAM( VALUE : TPARAM)
16724: Procedure AddPassword( const Password : string)
16725: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
16726: Procedure AddState( oState : TniRegularExpressionState)
16727: Procedure AddStrings( Strings : TStrings);
16728: procedure AddStrings(Strings: TStrings);
16729: Procedure AddStrings1( Strings : TWideStrings);
16730: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
16731: Procedure AddToRecentDocs( const Filename : string)
16732: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
16733: Procedure AllFunctionsList1Click( Sender : TObject)
16734: procedure AllObjectsList1Click(Sender: TObject);
16735: Procedure Allocate( AAllocateBytes : Integer)
16736: procedure AllResourceList1Click(Sender: TObject);
16737: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
16738: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
16739: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
16740: Procedure AnsiFree( var s : AnsiString)
16741: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
16742: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
16743: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
16744: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
16745: Procedure AntiFreeze;
16746: Procedure APPEND
16747: Procedure Append( const S : WideString)
16748: procedure Append(S: string);
16749: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
16750: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
16751: Procedure AppendChunk( Val : OleVariant)
16752: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
16753: Procedure AppendStr( var Dest : string; S : string)
16754: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
16755: Procedure ApplyRange
16756: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
16757: Procedure Arrange( Code : TListArrangement)
16758: procedure Assert(expr : Boolean; const msg: string);
16759: procedure Assert2(expr : Boolean; const msg: string);
16760: Procedure Assign( AList : TCustomBucketList)
16761: Procedure Assign( Other : TObject)
16762: Procedure Assign( Source : TDragObject)
16763: Procedure Assign( Source : TPersistent)
16764: Procedure Assign(Source: TPersistent)
16765: procedure Assign2(mystring, mypath: string);
16766: Procedure AssignCurValues( Source : TDataSet);
16767: Procedure AssignCurValues1( const CurValues : Variant);
16768: Procedure ASSIGNFIELD( FIELD : TFIELD)
16769: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
16770: Procedure AssignFile(var F: Text; FileName: string)
16771: procedure AssignFile(var F: TextFile; FileName: string)
16772: procedure AssignFileRead(var mystring, myfilename: string);
16773: procedure AssignFileWrite(mystring, myfilename: string);
16774: Procedure AssignTo( Other : TObject)
16775: Procedure AssignValues( Value : TParameters)
```

```
16776: Procedure ASSIGNVALUES( VALUE : TPARAMS)
16777: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
16778: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
16779: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
16780: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
16781: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
16782: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
16783: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
16784: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
16785: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
16786: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
16787: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
16788: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
16789: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
16790: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
16791: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
16792: procedure Beep
16793: Procedure BeepOk
16794: Procedure BeepQuestion
16795: Procedure BeepHand
16796: Procedure BeepExclamation
16797: Procedure BeepAsterisk
16798: Procedure BeepInformation
16799: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
16800: Procedure BeginLayout
16801: Procedure BeginTimer( const Delay, Resolution : Cardinal)
16802: Procedure BeginUpdate
16803: procedure BeginUpdate;
16804: procedure BigScreen1Click(Sender: TObject);
16805: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
16806: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
16807: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
16808: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
16809: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
16810: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
16811: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
16812: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
16813: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
16814: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
16815: Procedure BreakPointMenuClick( Sender : TObject)
16816: procedure BRINGTOFRONT
16817: procedure BringToFront;
16818: Procedure btnBackClick( Sender : TObject)
16819: Procedure btnBrowseClick( Sender : TObject)
16820: Procedure BtnClick( Index : TNavigateBtn)
16821: Procedure btnLargeIconsClick( Sender : TObject)
16822: Procedure BuildAndSendRequest( AURI : TIdURI)
16823: Procedure BuildCache
16824: Procedure BurnMemory( var Buff, BuffLen : integer)
16825: Procedure BurnMemoryStream( Destructo : TMemoryStream)
16826: Procedure CalculateFirstSet
16827: Procedure Cancel
16828: procedure CancelDrag;
16829: Procedure CancelEdit
16830: procedure CANCELHINT
16831: Procedure CancelRange
16832: Procedure CancelUpdates
16833: Procedure CancelWriteBuffer
16834: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
16835: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
16836: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
16837: procedure CaptureScreenFormat(vname: string; vextension: string);
16838: procedure CaptureScreenPNG(vname: string);
16839: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
16840: procedure CASCADE
16841: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
16842: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
16843: Procedure cbPathClick( Sender : TObject)
16844: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
16845: Procedure cedebugAfterExecute( Sender : TPSScript)
16846: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
16847: Procedure cedebugCompile( Sender : TPSScript)
16848: Procedure cedebugExecute( Sender : TPSScript)
16849: Procedure cedebugIdle( Sender : TObject)
16850: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
16851: Procedure CenterHeight( const pc, pcParent : TControl)
16852: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
16853: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
16854: Procedure Change
16855: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
16856: Procedure Changed
16857: Procedure ChangeDir( const ADirName : string)
16858: Procedure ChangeDirUp
16859: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
16860: Procedure ChangeLevelBy( Value : TChangeRange)
16861: Procedure ChDir(const s: string)
16862: Procedure Check(Status: Integer)
16863: Procedure CheckCommonControl( CC : Integer)
16864: Procedure CHECKFIELDNAME( const FIELDNAME : String)
```

```
16865: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
16866: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
16867: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
16868: Procedure CheckToken( T : Char)
16869: procedure CheckToken(t:char)
16870: Procedure CheckTokenSymbol( const S : string)
16871: procedure CheckTokenSymbol(s:string)
16872: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
16873: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
16874: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
16875: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
16876: procedure CipherFile1Click(Sender: TObject);
16877: Procedure Clear;
16878: Procedure Clear1Click( Sender : TObject)
16879: Procedure ClearColor( Color : TColor)
16880: Procedure CLEARITEM( AITEM : TMENUITEM)
16881: Procedure ClearMapping
16882: Procedure ClearSelection( KeepPrimary : Boolean)
16883: Procedure ClearWriteBuffer
16884: Procedure Click
16885: Procedure Close
16886: Procedure Close1Click( Sender : TObject)
16887: Procedure CloseDatabase( Database : TDatabase)
16888: Procedure CloseDataSets
16889: Procedure CloseDialog
16890: Procedure CloseFile(var F: Text);
16891: Procedure Closure
16892: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16893: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16894: Procedure CodeCompletionList1Click( Sender : TObject)
16895: Procedure ColEnter
16896: Procedure Collapse
16897: Procedure Collapse( Recurse : Boolean)
16898: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
16899: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
16900: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
16901: Procedure Compile1Click( Sender : TObject)
16902: procedure ComponentCount1Click(Sender: TObject);
16903: Procedure Compress(azipfolder, azipfile: string)
16904: Procedure DeCompress(azipfolder, azipfile: string)
16905: Procedure XZip(azipfolder, azipfile: string)
16906: Procedure XUnZip(azipfolder, azipfile: string)
16907: Procedure Connect( const ATimeout : Integer)
16908: Procedure Connect( Socket : TSocket)
16909: procedure Console1Click(Sender: TObject);
16910: Procedure Continue
16911: Procedure ContinueCount( var Counter : TJclCounter)
16912: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
16913: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
16914: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
16915: Procedure ConvertImage(vsource, vdestination: string);
16916: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
16917: Procedure ConvertToGray(Cnv: TCanvas);
16918: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
16919: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
16920: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
16921: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
16922: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
16923: Procedure CopyFrom( mbCopy : TMyBigInt)
16924: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
16925: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
16926: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
16927: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALen:Int)
16928: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
16929: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
16930: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
16931: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
16932: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
16933: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
16934: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
16935: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
16936: Procedure CopyToClipboard
16937: Procedure CountParts
16938: Procedure CreateDataSet
16939: Procedure CreateEmptyFile( const FileName : string)
16940: Procedure CreateFileFromString( const FileName, Data : string)
16941: Procedure CreateFromDelta( Source : TPacketDataSet)
16942: procedure CREATEHANDLE
16943: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
16944: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
16945: Procedure CreateTable
16946: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
16947: procedure CSyntax1Click(Sender: TObject);
16948: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
16949: Procedure CURSORPOSCHANGED
16950: procedure CutFirstDirectory(var S: String)
16951: Procedure DataBaseError(const Message: string)
```

```
16952: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
16953: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
16954: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
16955: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
16956: Procedure DBIError(errorCode: Integer)
16957: Procedure DebugOutput( const AText : string)
16958: Procedure DebugRun1Click( Sender : TObject)
16959: procedure Dec;
16960: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
16961: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
16962: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
16963: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
16964: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
16965: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
16966: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
16967: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
16968: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
16969: Procedure Decompile1Click( Sender : TObject)
16970: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
16971: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
16972: Procedure DeferLayout
16973: Procedure defFileread
16974: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
16975: Procedure DelayMicroseconds( const MicroSeconds : Integer)
16976: Procedure Delete
16977: Procedure Delete( const AFilename : string)
16978: Procedure Delete( const Index : Integer)
16979: Procedure DELETE( INDEX : INTEGER)
16980: Procedure Delete( Index : LongInt)
16981: Procedure Delete( Node : TTreeNode)
16982: procedure Delete(var s: AnyString; ifrom, icount: Longint);
16983: Procedure DeleteAlias( const Name : string)
16984: Procedure DeleteDriver( const Name : string)
16985: Procedure DeleteIndex( const Name : string)
16986: Procedure DeleteKey( const Section, Ident : String)
16987: Procedure DeleteRecords
16988: Procedure DeleteRecords( AffectRecords : TAffectRecords)
16989: Procedure DeleteString( var pStr : String; const pDelStr : string)
16990: Procedure DeleteTable
16991: procedure DelphiSite1Click(Sender: TObject);
16992: Procedure Deselect
16993: Procedure Deselect( Node : TTreeNode)
16994: procedure DestroyComponents
16995: Procedure DestroyHandle
16996: Procedure Diff( var X : array of Double)
16997: procedure Diff(var X: array of Double);
16998: procedure DISABLEALIGN
16999: Procedure DisableConstraints
17000: Procedure Disconnect
17001: Procedure Disconnect( Socket : TSocket)
17002: Procedure Dispose
17003: procedure Dispose(P: PChar)
17004: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
17005: Procedure DoKey( Key : TDBCtrlGridKey)
17006: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17007: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17008: Procedure Dormant
17009: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
17010: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
17011: Procedure DoubleToComp( Value : Double; var Result : Comp)
17012: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
17013: procedure Draw(X, Y: Integer; Graphic: TGraphic);
17014: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
17015: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17016: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
17017: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17018: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
17019: procedure DrawFocusRect(const Rect: TRect);
17020: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
17021: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17022: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
17023: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
17024: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
17025: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
17026: Procedure DropConnections
17027: Procedure DropDown
17028: Procedure DumpDescription( oStrings : TStrings)
17029: Procedure DumpStateTable( oStrings : TStrings)
17030: Procedure EDIT
17031: Procedure EditButtonClick
17032: Procedure EditFont1Click( Sender : TObject)
17033: procedure Ellipse(X1, Y1, X2, Y2: Integer);
17034: Procedure Ellipse1( const Rect : TRect);
17035: Procedure EMMS
17036: Procedure Encode( ADest : TStream)
17037: procedure ENDDRAG(DROP:BOOLEAN)
17038: Procedure EndEdit( Cancel : Boolean)
```

```
17039: Procedure EndTimer
17040: Procedure EndUpdate
17041: Procedure EraseSection( const Section : string)
17042: Procedure Error( const Ident : string)
17043: procedure Error(Ident:Integer)
17044: Procedure ErrorFmt( const Ident : string; const Args : array of const)
17045: Procedure ErrorStr( const Message : string)
17046: procedure ErrorStr(Message:String)
17047: Procedure Exchange( Index1, Index2 : Integer)
17048: procedure Exchange(Index1, Index2: Integer);
17049: Procedure Exec( FileName, Parameters, Directory : string)
17050: Procedure ExecProc
17051: Procedure ExecSQL( UpdateKind : TUpdateKind)
17052: Procedure Execute
17053: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
17054: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
17055: Procedure ExecuteCommand(executeFile, paramstring: string)
17056: Procedure ExecuteShell(executeFile, paramstring: string)
17057: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
17058: Procedure ExitThread(ExitCode: Integer); stdcall;
17059: Procedure ExitProcess(ExitCode: Integer); stdcall;
17060: Procedure Expand( AUserName : String; AResults : TStrings)
17061: Procedure Expand( Recurse : Boolean)
17062: Procedure ExportClipboard1Click( Sender : TObject)
17063: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
17064: Procedure ExtractContentFields( Strings : TStrings)
17065: Procedure ExtractCookieFields( Strings : TStrings)
17066: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
17067: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
17068: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
17069: Procedure ExtractQueryFields( Strings : TStrings)
17070: Procedure FastDegToGrad
17071: Procedure FastDegToRad
17072: Procedure FastGradToDeg
17073: Procedure FastGradToRad
17074: Procedure FastRadToDeg
17075: Procedure FastRadToGrad
17076: Procedure FileClose( Handle : Integer)
17077: Procedure FileClose(handle: integer)
17078: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
17079: Procedure FileStructure( AStructure : TIdFTPDataStructure)
17080: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
17081: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
17082: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
17083: Procedure FillChar2(var X: PChar ; count: integer; value: char)
17084: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
17085: Procedure FillIPList
17086: procedure FillRect(const Rect: TRect);
17087: Procedure FillTStrings( AStrings : TStrings)
17088: Procedure FilterOnBookmarks( Bookmarks : array of const)
17089: procedure FinalizePackage(Module: HMODULE)
17090: procedure FindClose;
17091: procedure FindClose2(var F: TSearchRec)
17092: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
17093: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
17094: Procedure FindNearest( const KeyValues : array of const)
17095: Procedure FinishContext
17096: Procedure FIRST
17097: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
17098: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
17099: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
17100: Procedure FlushSchemaCache( const TableName : string)
17101: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
17102: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
17103: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
17104: Procedure FormActivate( Sender : TObject)
17105: procedure FormatLn(const format: String; const args: array of const);  //alias
17106: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
17107: Procedure FormCreate( Sender : TObject)
17108: Procedure FormDestroy( Sender : TObject)
17109: Procedure FormKeyPress( Sender : TObject; var Key : Char)
17110: procedure FormOutput1Click(Sender: TObject);
17111: Procedure FormToHtml( Form : TForm; Path : string)
17112: procedure FrameRect(const Rect: TRect);
17113: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
17114: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
17115: Procedure Free( Buffer : TRecordBuffer)
17116: Procedure Free( Buffer : TValueBuffer)
17117: Procedure Free;
17118: Procedure FreeAndNil(var Obj:TObject)
17119: Procedure FreeImage
17120: procedure FreeMem(P: PChar; Size: Integer)
17121: Procedure FreeTreeData( Tree : TUpdateTree)
17122: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
17123: Procedure FullCollapse
17124: Procedure FullExpand
17125: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
```

```
17126: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
17127: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
17128: Procedure Get1( AURL : string; const AResponseContent : TStream);
17129: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
17130: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
17131: Procedure GetAliasNames( List : TStrings)
17132: Procedure GetAliasParams( const AliasName : string; List : TStrings)
17133: Procedure GetApplicationsRunning( Strings : TStrings)
17134: Procedure GetCommandTypes( List : TWideStrings)
17135: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
17136: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
17137: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
17138: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
17139: Procedure GetDatabaseNames( List : TStrings)
17140: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
17141: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
17142: Procedure GetDir(d: byte; var s: string)
17143: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
17144: Procedure GetDriverNames( List : TStrings)
17145: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
17146: Procedure GetDriverParams( const DriverName : string; List : TStrings)
17147: Procedure GetEMails1Click( Sender : TObject)
17148: Procedure getEnvironmentInfo;
17149: Function getEnvironmentString: string;
17150: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
17151: Procedure GetFieldNames( const TableName : string; List : TStrings)
17152: Procedure GetFieldNames( const TableName : string; List : TStrings);
17153: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
17154: Procedure GETFIELDNAMES( LIST : TSTRINGS)
17155: Procedure GetFieldNames1( const TableName : string; List : TStrings);
17156: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
17157: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
17158: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
17159: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
17160: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
17161: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
17162: Procedure GetFormatSettings
17163: Procedure GetFromDIB( var DIB : TBitmapInfo)
17164: Procedure GetFromHDIB( HDIB : HBitmap)
17165: Procedure GetIcon( Index : Integer; Image : TIcon);
17166: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
17167: Procedure GetIndexInfo( IndexName : string)
17168: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
17169: Procedure GetIndexNames( List : TStrings)
17170: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
17171: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
17172: Procedure GetIndexNames4( const TableName : string; List : TStrings);
17173: Procedure GetInternalResponse
17174: Procedure GETITEMNAMES( LIST : TSTRINGS)
17175: procedure GetMem(P: PChar; Size: Integer)
17176: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
17177: procedure GetPackageDescription(ModuleName: PChar): string)
17178: Procedure GetPackageNames( List : TStrings);
17179: Procedure GetPackageNames1( List : TWideStrings);
17180: Procedure GetParamList( List : TList; const ParamNames : WideString)
17181: Procedure GetProcedureNames( List : TStrings);
17182: Procedure GetProcedureNames( List : TWideStrings);
17183: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
17184: Procedure GetProcedureNames1( List : TStrings);
17185: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
17186: Procedure GetProcedureNames3( List : TWideStrings);
17187: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
17188: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
17189: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
17190: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
17191: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
17192: Procedure GetProviderNames( Names : TWideStrings);
17193: Procedure GetProviderNames( Proc : TGetStrProc)
17194: Procedure GetProviderNames1( Names : TStrings);
17195: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
17196: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no autoopen
       image
17197: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
17198: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
17199: Procedure GetSchemaNames( List : TStrings);
17200: Procedure GetSchemaNames1( List : TWideStrings);
17201: Procedure GetSessionNames( List : TStrings)
17202: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
17203: Procedure GetStrings( List : TStrings)
17204: Procedure GetSystemTime; stdcall;
17205: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
17206: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
17207: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
17208: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
17209: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
17210: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
17211: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
17212: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
```

```
17213: Procedure GetVisibleWindows( List : Tstrings)
17214: Procedure GoBegin
17215: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
17216: Procedure GotoCurrent( Table : TTable)
17217: procedure GotoEnd1Click(Sender: TObject);
17218: Procedure GotoNearest
17219: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
17220: Procedure HandleException( E : Exception; var Handled : Boolean)
17221: procedure HANDLEMESSAGE
17222: procedure HandleNeeded;
17223: Procedure Head( AURL : string)
17224: Procedure Help( var AHelpContents : TStringList; ACommand : String)
17225: Procedure HexToBinary( Stream : TStream)
17226: procedure HexToBinary(Stream:TStream)
17227: Procedure HideDragImage
17228: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
17229: Procedure HideTraybar
17230: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
17231: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
17232: Procedure HookOSExceptions
17233: Procedure HookSignal( RtlSigNum : Integer)
17234: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
17235: Procedure HTMLSyntax1Click( Sender : TObject)
17236: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
17237: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
17238: Procedure ImportfromClipboard1Click( Sender : TObject)
17239: Procedure ImportfromClipboard2Click( Sender : TObject)
17240: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
17241: procedure Incb(var x: byte);
17242: Procedure Include1Click( Sender : TObject)
17243: Procedure IncludeOFF;  //preprocessing
17244: Procedure IncludeON;
17245: procedure Info1Click(Sender: TObject);
17246: Procedure InitAltRecBuffers( CheckModified : Boolean)
17247: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
17248: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
17249: Procedure InitData( ASource : TDataSet)
17250: Procedure InitDelta( ADelta : TPacketDataSet);
17251: Procedure InitDelta1( const ADelta : OleVariant);
17252: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
17253: Procedure Initialize
17254: procedure InitializePackage(Module: HMODULE)
17255: Procedure INITIATEACTION
17256: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
17257: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
17258: Procedure InitModule( AModule : TComponent)
17259: Procedure InitStdConvs
17260: Procedure InitTreeData( Tree : TUpdateTree)
17261: Procedure INSERT
17262: Procedure Insert( Index : Integer; AClass : TClass)
17263: Procedure Insert( Index : Integer; AComponent : TComponent)
17264: Procedure Insert( Index : Integer; AObject : TObject)
17265: Procedure Insert( Index : Integer; const S : WideString)
17266: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
17267: Procedure Insert(Index: Integer; const S: string);
17268: procedure Insert(Index: Integer; S: string);
17269: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
17270: procedure InsertComponent(AComponent:TComponent)
17271: procedure InsertControl(AControl: TControl);
17272: Procedure InsertIcon( Index : Integer; Image : TIcon)
17273: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
17274: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
17275: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
17276: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
17277: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
17278: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
17279: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
17280: Procedure InternalBeforeResolve( Tree : TUpdateTree)
17281: Procedure InvalidateModuleCache
17282: Procedure InvalidateTitles
17283: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
17284: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
17285: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
17286: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
17287: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
17288: procedure JavaSyntax1Click(Sender: TObject);
17289: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
17290: Procedure KillDataChannel
17291: Procedure Largefont1Click( Sender : TObject)
17292: Procedure LAST
17293: Procedure LaunchCpl( FileName : string)
17294: Procedure Launch( const AFile : string)
17295: Procedure LaunchFile( const AFile : string)
17296: Procedure LetFileList(FileList: TStringlist; apath: string);
17297: Procedure lineToNumber( xmemo : String; met : boolean)
17298: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
       DefaultDraw:Bool)
```

```
17299: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
17300: Procedure ListViewData( Sender : TObject; Item : TListItem)
17301: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
17302: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
17303: Procedure ListViewDblClick( Sender : TObject)
17304: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
17305: Procedure ListDLLExports(const FileName: string; List: TStrings);
17306: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
17307: procedure LoadBytecode1Click(Sender: TObject);
17308: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
17309: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
17310: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
17311: Procedure LoadFromFile( AFileName : string)
17312: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
17313: Procedure LoadFromFile( const FileName : string)
17314: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
17315: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
17316: Procedure LoadFromFile( const FileName : WideString)
17317: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
17318: Procedure LoadFromFile(const AFileName: string)
17319: procedure LoadFromFile(FileName: string);
17320: procedure LoadFromFile(FileName:String)
17321: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
17322: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
17323: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
17324: Procedure LoadFromStream( const Stream : TStream)
17325: Procedure LoadFromStream( S : TStream)
17326: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
17327: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
17328: Procedure LoadFromStream( Stream : TStream)
17329: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
17330: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
17331: procedure LoadFromStream(Stream: TStream);
17332: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
17333: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
17334: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
17335: Procedure LoadLastFile1Click( Sender : TObject)
17336: { LoadIcoToImage loads two icons from resource named NameRes,
17337:   into two image lists ALarge and ASmall}
17338: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
17339: Procedure LoadMemo
17340: Procedure LoadParamsFromIniFile( FFileName : WideString)
17341: Procedure Lock
17342: Procedure Login
17343: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
17344: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
17345: Procedure MakeCaseInsensitive
17346: Procedure MakeDeterministic( var bChanged : boolean)
17347: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
17348: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
17349: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
17350: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
17351: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
17352: Procedure SetRectComplexFormatStr( const S : string)
17353: Procedure SetPolarComplexFormatStr( const S : string)
17354: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
17355: Procedure MakeVisible
17356: Procedure MakeVisible( PartialOK : Boolean)
17357: Procedure Manual1Click( Sender : TObject)
17358: Procedure MarkReachable
17359: Procedure maXbox;  //shows the exe version data in a win box
17360: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
17361: Procedure Memo1Change( Sender : TObject)
17362: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
17363: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
17364: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
17365: procedure Memory1Click(Sender: TObject);
17366: Procedure MERGE( MENU : TMAINMENU)
17367: Procedure MergeChangeLog
17368: procedure MINIMIZE
17369: Procedure MinimizeMaxbox;
17370: Procedure MkDir(const s: string)
17371: Procedure mnuPrintFont1Click( Sender : TObject)
17372: procedure ModalStarted
17373: Procedure Modified
17374: Procedure ModifyAlias( Name : string; List : TStrings)
17375: Procedure ModifyDriver( Name : string; List : TStrings)
17376: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
17377: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
17378: Procedure Move( CurIndex, NewIndex : Integer)
17379: procedure Move(CurIndex, NewIndex: Integer);
17380: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
17381: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
17382: Procedure moveCube( o : TMyLabel)
17383: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
17384: procedure MoveTo(X, Y: Integer);
```

```
17385: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
17386: Procedure MovePoint(var x,y:Extended; const angle:Extended);
17387: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
17388: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
17389: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
17390: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
17391: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
17392: procedure New(P: PChar)
17393: procedure New1Click(Sender: TObject);
17394: procedure NewInstance1Click(Sender: TObject);
17395: Procedure NEXT
17396: Procedure NextMonth
17397: Procedure Noop
17398: Procedure NormalizePath( var APath : string)
17399: procedure ObjectBinaryToText(Input, Output: TStream)
17400: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17401: procedure ObjectResourceToText(Input, Output: TStream)
17402: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17403: procedure ObjectTextToBinary(Input, Output: TStream)
17404: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17405: procedure ObjectTextToResource(Input, Output: TStream)
17406: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17407: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
17408: Procedure Open( const UserID : WideString; const Password : WideString);
17409: Procedure Open;
17410: Procedure open1Click( Sender : TObject)
17411: Procedure OpenCdDrive
17412: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
17413: Procedure OpenCurrent
17414: Procedure OpenFile(vfilenamepath: string)
17415: Procedure OpenDirectory1Click( Sender : TObject)
17416: Procedure OpenIndexFile( const IndexName : string)
17417: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemID:OleVariant;DataSet:TADODataSet)
17418: Procedure OpenWriteBuffer( const AThreshhold : Integer)
17419: Procedure OptimizeMem
17420: Procedure Options1( AURL : string);
17421: Procedure OutputDebugString(lpOutputString : PChar)
17422: Procedure PackBuffer
17423: Procedure Paint
17424: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
17425: Procedure PaintToTBitmap( Target : TBitmap)
17426: Procedure PaletteChanged
17427: Procedure ParentBiDiModeChanged
17428: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
17429: Procedure PasteFromClipboard;
17430: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
17431: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
17432: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
17433: Procedure PError( Text : string)
17434: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17435: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
17436: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
17437: procedure playmp3(mpath: string);
17438: Procedure PlayMP31Click( Sender : TObject)
17439: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
17440: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
17441: procedure PolyBezier(const Points: array of TPoint);
17442: procedure PolyBezierTo(const Points: array of TPoint);
17443: procedure Polygon(const Points: array of TPoint);
17444: procedure Polyline(const Points: array of TPoint);
17445: Procedure Pop
17446: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
17447: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
17448: Procedure POPUP( X, Y : INTEGER)
17449: Procedure PopupURL(URL : WideString);
17450: Procedure POST
17451: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
17452: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
17453: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
17454: Procedure PostUser( const Email, FirstName, LastName : WideString)
17455: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
17456: procedure Pred(X: int64);
17457: Procedure Prepare
17458: Procedure PrepareStatement
17459: Procedure PreProcessXML( AList : TStrings)
17460: Procedure PreventDestruction
17461: Procedure Print( const Caption : string)
17462: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
17463: procedure printf(const format: String; const args: array of const);
17464: Procedure PrintList(Value: TStringList);
17465: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
17466: Procedure Printout1Click( Sender : TObject)
17467: Procedure ProcessHeaders
17468: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
17469: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
17470: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
17471: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
17472: Procedure ProcessMessagesOFF;  //application.processmessages
```

```
17473: Procedure ProcessMessagesON;
17474: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
17475: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
17476: Procedure Proclist Size is: 3797 /1415
17477: Procedure procMessClick( Sender : TObject)
17478: Procedure PSScriptCompile( Sender : TPSScript)
17479: Procedure PSScriptExecute( Sender : TPSScript)
17480: Procedure PSScriptLine( Sender : TObject)
17481: Procedure Push( ABoundary : string)
17482: procedure PushItem(AItem: Pointer)
17483: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
17484: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
17485: procedure PutLinuxLines(const Value: string)
17486: Procedure Quit
17487: Procedure RaiseConversionError( const AText : string);
17488: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
17489: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
17490: procedure RaiseException(Ex: TIFException; Param: String);
17491: Procedure RaiseExceptionForLastCmdResult;
17492: procedure RaiseLastException;
17493: procedure RaiseException2;
17494: Procedure RaiseLastOSError
17495: Procedure RaiseLastWin32;
17496: procedure RaiseLastWin32Error)
17497: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
17498: Procedure RandomFillStream( Stream : TMemoryStream)
17499: procedure randomize;
17500: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
17501: Procedure RCS
17502: Procedure Read( Socket : TSocket)
17503: Procedure ReadBlobData
17504: procedure ReadBuffer(Buffer:String;Count:LongInt)
17505: procedure ReadOnly1Click(Sender: TObject);
17506: Procedure ReadSection( const Section : string; Strings : TStrings)
17507: Procedure ReadSections( Strings : TStrings)
17508: Procedure ReadSections( Strings : TStrings);
17509: Procedure ReadSections1( const Section : string; Strings : TStrings);
17510: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
17511: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
17512: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
17513: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
17514: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
17515: Procedure Realign;
17516: procedure Rectangle(X1, Y1, X2, Y2: Integer);
17517: Procedure Rectangle1( const Rect : TRect);
17518: Procedure RectCopy( var Dest : TRect; const Source : TRect)
17519: Procedure RectFitToScreen( var R : TRect)
17520: Procedure RectGrow( var R : TRect; const Delta : Integer)
17521: Procedure RectGrowX( var R : TRect; const Delta : Integer)
17522: Procedure RectGrowY( var R : TRect; const Delta : Integer)
17523: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
17524: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
17525: Procedure RectNormalize( var R : TRect)
17526: //  TFileCallbackProcedure = procedure(filename:string);
17527: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
17528: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
17529: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
17530: Procedure Refresh;
17531: Procedure RefreshData( Options : TFetchOptions)
17532: Procedure REFRESHLOOKUPLIST
17533: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
17534: Procedure RegisterChanges( Value : TChangeLink)
17535: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
17536: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
17537: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
17538: Procedure ReInitialize( ADelay : Cardinal)
17539: procedure RELEASE
17540: Procedure Remove( const AByteCount : integer)
17541: Procedure REMOVE( FIELD : TFIELD)
17542: Procedure REMOVE( ITEM : TMENUITEM)
17543: Procedure REMOVE( POPUP : TPOPUPMENU)
17544: Procedure RemoveAllPasswords
17545: procedure RemoveComponent(AComponent:TComponent)
17546: Procedure RemoveDir( const ADirName : string)
17547: Procedure RemoveLambdaTransitions( var bChanged : boolean)
17548: Procedure REMOVEPARAM( VALUE : TPARAM)
17549: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
17550: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
17551: Procedure Rename( const ASourceFile, ADestFile : string)
17552: Procedure Rename( const FileName : string; Reload : Boolean)
17553: Procedure RenameTable( const NewTableName : string)
17554: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
17555: Procedure Replace1Click( Sender : TObject)
17556: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
17557: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
17558: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
17559: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
17560: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
17561: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
```

```
17562: Procedure Requery( Options : TExecuteOptions)
17563: Procedure Reset
17564: Procedure Reset1Click( Sender : TObject)
17565: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
17566: procedure ResourceExplore1Click(Sender: TObject);
17567: Procedure RestoreContents
17568: Procedure RestoreDefaults
17569: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
17570: Procedure RetrieveHeaders
17571: Procedure RevertRecord
17572: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
17573: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17574: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
17575: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
17576: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
17577: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
17578: Procedure RleCompress2( Stream : TStream)
17579: Procedure RleDecompress2( Stream : TStream)
17580: Procedure RmDir(const S: string)
17581: Procedure Rollback
17582: Procedure Rollback( TransDesc : TTransactionDesc)
17583: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
17584: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
17585: Procedure RollbackTrans
17586: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
17587: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
17588: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
17589: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
17590: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
17591: Procedure S_EBox( const AText : string)
17592: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
17593: Procedure S_IBox( const AText : string)
17594: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
17595: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
17596: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
17597: Procedure SampleVarianceAndMean
17598: ( const X : TDynFloatArray; var Variance, Mean : Float)
17599: Procedure Save2Click( Sender : TObject)
17600: Procedure Saveas3Click( Sender : TObject)
17601: Procedure Savebefore1Click( Sender : TObject)
17602: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
17603: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
17604: Procedure SaveConfigFile
17605: Procedure SaveOutput1Click( Sender : TObject)
17606: procedure SaveScreenshotClick(Sender: TObject);
17607: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
17608: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
17609: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
17610: Procedure SaveToFile( AFileName : string)
17611: Procedure SAVETOFILE( const FILENAME : String)
17612: Procedure SaveToFile( const FileName : WideString)
17613: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
17614: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
17615: procedure SaveToFile(FileName: string);
17616: procedure SaveToFile(FileName:String)
17617: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
17618: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
17619: Procedure SaveToStream( S : TStream)
17620: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
17621: Procedure SaveToStream( Stream : TStream)
17622: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
17623: procedure SaveToStream(Stream: TStream);
17624: procedure SaveToStream(Stream:TStream)
17625: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
17626: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
17627: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
17628: procedure Say(const sText: string)
17629: Procedure SBytecode1Click( Sender : TObject)
17630: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
17631: procedure ScriptExplorer1Click(Sender: TObject);
17632: Procedure Scroll( Distance : Integer)
17633: Procedure Scroll( DX, DY : Integer)
17634: procedure ScrollBy(DeltaX, DeltaY: Integer);
17635: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
17636: Procedure ScrollTabs( Delta : Integer)
17637: Procedure Search1Click( Sender : TObject)
17638: procedure SearchAndOpenDoc(vfilenamepath: string)
17639: procedure SearchAndOpenFile(vfilenamepath: string)
17640: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
17641: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
17642: Procedure SearchNext1Click( Sender : TObject)
17643: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
17644: Procedure Select1( const Nodes : array of TTreeNode);
17645: Procedure Select2( Nodes : TList);
17646: Procedure SelectNext( Direction : Boolean)
17647: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
17648: Procedure SelfTestPEM  //unit uPSI_cPEM
17649: Procedure Send( AMsg : TIdMessage)
```

```
17650: //config forst in const MAILINIFILE = 'maildef.ini';
17651: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
17652: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
17653: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
17654: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
17655: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
17656: Procedure SendResponse
17657: Procedure SendStream( AStream : TStream)
17658: Procedure Set8087CW( NewCW : Word)
17659: Procedure SetAll( One, Two, Three, Four : Byte)
17660: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
17661: Procedure SetAppDispatcher( const ADispatcher : TComponent)
17662: procedure SetArrayLength;
17663: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
17664: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
17665: Procedure SetAsHandle( Format : Word; Value : THandle)
17666: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
17667: procedure SetCaptureControl(Control: TControl);
17668: Procedure SetColumnAttributes
17669: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
17670: Procedure SetCustomHeader( const Name, Value : string)
17671: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
17672: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
17673: Procedure SetFocus
17674: procedure SetFocus; virtual;
17675: Procedure SetInitialState
17676: Procedure SetKey
17677: procedure SetLastError(ErrorCode: Integer)
17678: procedure SetLength;
17679: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
17680: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
17681: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
17682: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
17683: Procedure SetParams1( UpdateKind : TUpdateKind);
17684: Procedure SetPassword( const Password : string)
17685: Procedure SetPointer( Ptr : Pointer; Size : Longint)
17686: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
17687: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
17688: Procedure SetProvider( Provider : TComponent)
17689: Procedure SetProxy( const Proxy : string)
17690: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
17691: Procedure SetRange( const StartValues, EndValues : array of const)
17692: Procedure SetRangeEnd
17693: Procedure SetRate( const aPercent, aYear : integer)
17694: procedure SetRate(const aPercent, aYear: integer)
17695: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
17696: Procedure SetSafeCallExceptionMsg( Msg : String)
17697: procedure SETSELTEXTBUF(BUFFER:PCHAR)
17698: Procedure SetSize( AWidth, AHeight : Integer)
17699: procedure SetSize(NewSize:LongInt)
17700: procedure SetString(var s: string; buffer: PChar; len: Integer)
17701: Procedure SetStrings( List : TStrings)
17702: Procedure SetText( Text : PwideChar)
17703: procedure SetText(Text: PChar);
17704: Procedure SetTextBuf( Buffer : PChar)
17705: procedure SETTEXTBUF(BUFFER:PCHAR)
17706: Procedure SetTick( Value : Integer)
17707: Procedure SetTimeout( ATimeOut : Integer)
17708: Procedure SetTraceEvent( Event : TDBXTraceEvent)
17709: Procedure SetUserName( const UserName : string)
17710: Procedure SetWallpaper( Path : string);
17711: procedure ShellStyle1Click(Sender: TObject);
17712: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
17713: Procedure ShowFileProperties( const FileName : string)
17714: Procedure ShowInclude1Click( Sender : TObject)
17715: Procedure ShowInterfaces1Click( Sender : TObject)
17716: Procedure ShowLastException1Click( Sender : TObject)
17717: Procedure ShowMessage( const Msg : string)
17718: Procedure ShowMessageBig(const aText : string);
17719: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
17720: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
17721: Procedure MsgBig(const aText : string);        //alias
17722: procedure showmessage(mytext: string);
17723: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
17724: procedure ShowMessageFmt(const Msg: string; Params: array of const))
17725: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
17726: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
17727: Procedure ShowSearchDialog( const Directory : string)
17728: Procedure ShowSpecChars1Click( Sender : TObject)
17729: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
17730: Procedure ShredFile( const FileName : string; Times : Integer)
17731: procedure Shuffle(vQ: TStringList);
17732: Procedure ShuffleList( var List : array of Integer; Count : Integer)
17733: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
17734: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
17735: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
17736: Procedure Site( const ACommand : string)
17737: Procedure SkipEOL
```

```
17738: Procedure Sleep( ATime : cardinal)
17739: Procedure Sleep( milliseconds : Cardinal)
17740: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
17741: Procedure Slinenumbers1Click( Sender : TObject)
17742: Procedure Sort
17743: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
17744: procedure Speak(const sText: string)   //async like voice
17745: procedure Speak2(const sText: string)  //sync
17746: procedure Split(Str: string;  SubStr: string; List: TStrings);
17747: Procedure SplitNameValue( const Line : string; var Name, Value : string)
17748: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
17749: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
17750: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
17751: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
17752: procedure SQLSyntax1Click(Sender: TObject);
17753: Procedure SRand( Seed : RNG_IntType)
17754: Procedure Start
17755: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
17756: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
17757: Procedure StartTransaction( TransDesc : TTransactionDesc)
17758: Procedure Status( var AStatusList : TStringList)
17759: Procedure StatusBar1DblClick( Sender : TObject)
17760: Procedure StepInto1Click( Sender : TObject)
17761: Procedure StepIt
17762: Procedure StepOut1Click( Sender : TObject)
17763: Procedure Stop
17764: procedure stopmp3;
17765: procedure StartWeb(aurl: string);
17766: Procedure Str(aint: integer; astr: string); //of system
17767: Procedure StrDispose( Str : PChar)
17768: procedure StrDispose(Str: PChar)
17769: Procedure StrReplace(var Str: String; Old, New: String);
17770: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
17771: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
17772: Procedure StringToBytes( Value : String; Bytes : array of byte)
17773: procedure StrSet(c : Char; I : Integer; var s : String);
17774: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
17775: Procedure StructureMount( APath : String)
17776: procedure STYLECHANGED(SENDER:TOBJECT)
17777: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
17778: procedure Succ(X: int64);
17779: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
17780: procedure SwapChar(var X,Y: char);  //swapX follows
17781: Procedure SwapFloats( var X, Y : Float)
17782: procedure SwapGrid(grd: TStringGrid);
17783: Procedure SwapOrd( var I, J : Byte);
17784: Procedure SwapOrd( var X, Y : Integer)
17785: Procedure SwapOrd1( var I, J : Shortint);
17786: Procedure SwapOrd2( var I, J : Smallint);
17787: Procedure SwapOrd3( var I, J : Word);
17788: Procedure SwapOrd4( var I, J : Integer);
17789: Procedure SwapOrd5( var I, J : Cardinal);
17790: Procedure SwapOrd6( var I, J : Int64);
17791: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
17792: Procedure Synchronize1( Method : TMethod);
17793: procedure SyntaxCheck1Click(Sender: TObject);
17794: Procedure SysFreeString(const S: WideString); stdcall;
17795: Procedure TakeOver( Other : TLinearBitmap)
17796: Procedure Talkln(const sText: string)  //async voice
17797: procedure tbtn6resClick(Sender: TObject);
17798: Procedure tbtnUseCaseClick( Sender : TObject)
17799: procedure TerminalStyle1Click(Sender: TObject);
17800: Procedure Terminate
17801: Procedure texSyntax1Click( Sender : TObject)
17802: procedure TextOut(X, Y: Integer; Text: string);
17803: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
17804: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
17805: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
17806: Procedure TextStart
17807: procedure TILE
17808: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
17809: Procedure TitleClick( Column : TColumn)
17810: Procedure ToDo
17811: procedure toolbtnTutorialClick(Sender: TObject);
17812: Procedure Trace1( AURL : string; const AResponseContent : TStream);
17813: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
17814: Procedure Truncate
17815: procedure Tutorial101Click(Sender: TObject);
17816: procedure Tutorial10Statistics1Click(Sender: TObject);
17817: procedure Tutorial11Forms1Click(Sender: TObject);
17818: procedure Tutorial12SQL1Click(Sender: TObject);
17819: Procedure tutorial1Click( Sender : TObject)
17820: Procedure tutorial21Click( Sender : TObject)
17821: Procedure tutorial31Click( Sender : TObject)
17822: Procedure tutorial4Click( Sender : TObject)
17823: Procedure Tutorial5Click( Sender : TObject)
17824: procedure Tutorial6Click(Sender: TObject);
17825: procedure Tutorial91Click(Sender: TObject);
17826: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
```

```
17827: procedure UniqueString(var str: AnsiString)
17828: procedure UnloadLoadPackage(Module: HMODULE)
17829: Procedure Unlock
17830: Procedure UNMERGE( MENU : TMAINMENU)
17831: Procedure UnRegisterChanges( Value : TChangeLink)
17832: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
17833: Procedure UnregisterConversionType( const AType : TConvType)
17834: Procedure UnRegisterProvider( Prov : TCustomProvider)
17835: Procedure UPDATE
17836: Procedure UpdateBatch( AffectRecords : TAffectRecords)
17837: Procedure UPDATECURSORPOS
17838: Procedure UpdateFile
17839: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
17840: Procedure UpdateResponse( AResponse : TWebResponse)
17841: Procedure UpdateScrollBar
17842: Procedure UpdateView1Click( Sender : TObject)
17843: procedure Val(const s: string; var n, z: Integer)
17844: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
17845: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
17846: Procedure VariantAdd( const src : Variant; var dst : Variant)
17847: Procedure VariantAnd( const src : Variant; var dst : Variant)
17848: Procedure VariantArrayRedim( var V : Variant; High : Integer)
17849: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
17850: Procedure VariantClear( var V : Variant)
17851: Procedure VariantCpy( const src : Variant; var dst : Variant)
17852: Procedure VariantDiv( const src : Variant; var dst : Variant)
17853: Procedure VariantMod( const src : Variant; var dst : Variant)
17854: Procedure VariantMul( const src : Variant; var dst : Variant)
17855: Procedure VariantOr( const src : Variant; var dst : Variant)
17856: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
17857: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
17858: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
17859: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
17860: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
17861: Procedure VariantShl( const src : Variant; var dst : Variant)
17862: Procedure VariantShr( const src : Variant; var dst : Variant)
17863: Procedure VariantSub( const src : Variant; var dst : Variant)
17864: Procedure VariantXor( const src : Variant; var dst : Variant)
17865: Procedure VarCastError;
17866: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
17867: Procedure VarInvalidOp
17868: Procedure VarInvalidNullOp
17869: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
17870: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
17871: Procedure VarArrayCreateError
17872: Procedure VarResultCheck( AResult : HRESULT);
17873: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
17874: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
17875: Function VarTypeAsText( const AType : TVarType) : string
17876: procedure Voice(const sText: string) //async
17877: procedure Voice2(const sText: string) //sync
17878: Procedure WaitMiliSeconds( AMSec : word)
17879: Procedure WideAppend( var dst : WideString; const src : WideString)
17880: Procedure WideAssign( var dst : WideString; var src : WideString)
17881: Procedure WideDelete( var dst : WideString; index, count : Integer)
17882: Procedure WideFree( var s : WideString)
17883: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
17884: Procedure WideFromPChar( var dst : WideString; src : PChar)
17885: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
17886: Procedure WideSetLength( var dst : WideString; len : Integer)
17887: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
17888: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
17889: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
17890: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
17891: Procedure HttpGet(const Url: string; Stream:TStream);
17892: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
17893: Procedure WordWrap1Click( Sender : TObject)
17894: Procedure Write( const AOut : string)
17895: Procedure Write( Socket : TSocket)
17896: procedure Write(S: string);
17897: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
17898: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
17899: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
17900: procedure WriteBuffer(Buffer:String;Count:LongInt)
17901: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
17902: Procedure WriteChar( AValue : Char)
17903: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
17904: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
17905: Procedure WriteFloat( const Section, Name : string; Value : Double)
17906: Procedure WriteHeader( AHeader : TStrings)
17907: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
17908: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
17909: Procedure WriteLn( const AOut : string)
17910: procedure Writeln(s: string);
17911: Procedure WriteLog( const FileName, LogLine : string)
17912: Procedure WriteRFCReply( AReply : TIdRFCReply)
17913: Procedure WriteRFCStrings( AStrings : TStrings)
17914: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
17915: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
```

```
17916: Procedure WriteString( const Section, Ident, Value : String)
17917: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
17918: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
17919: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
17920: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
17921: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
17922: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
17923: procedure XMLSyntax1Click(Sender: TObject);
17924: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
17925: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
17926: Procedure ZeroFillStream( Stream : TMemoryStream)
17927: procedure XMLSyntax1Click(Sender: TObject);
17928: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
17929: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
17930: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
17931: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
17932: procedure(Sender, Source: TObject;X, Y: Integer)
17933: procedure(Sender, Target: TObject; X, Y: Integer)
17934: procedure(Sender: TObject; ASection, AWidth: Integer)
17935: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
17936: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
17937: procedure(Sender: TObject; var Action: TCloseAction)
17938: procedure(Sender: TObject; var CanClose: Boolean)
17939: procedure(Sender: TObject; var Key: Char);
17940: ProcedureName ProcedureNames ProcedureParametersCursor @
17941:
17942: *************Now Constructors constructor *************
17943:  Size is: 1248 1115 996 628 550 544 501 459 (381)
17944:  Attach( VersionInfoData : Pointer; Size : Integer)
17945:  constructor Create( ABuckets : TBucketListSizes)
17946:  Create( ACallBackWnd : HWND)
17947:  Create( AClient : TCustomTaskDialog)
17948:  Create( AClient : TIdTelnet)
17949:  Create( ACollection : TCollection)
17950:  Create( ACollection : TFavoriteLinkItems)
17951:  Create( ACollection : TTaskDialogButtons)
17952:  Create( AConnection : TIdCustomHTTP)
17953:  Create( ACreateSuspended : Boolean)
17954:  Create( ADataSet : TCustomSQLDataSet)
17955:  CREATE( ADATASET : TDATASET)
17956:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
17957:  Create( AGrid : TCustomDBGrid)
17958:  Create( AGrid : TStringGrid; AIndex : Longint)
17959:  Create( AHTTP : TIdCustomHTTP)
17960:  Create( AListItems : TListItems)
17961:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
17962:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
17963:  Create( AOwner : TCommonCalendar)
17964:  Create( AOwner : TComponent)
17965:  CREATE( AOWNER : TCOMPONENT)
17966:  Create( AOwner : TCustomListView)
17967:  Create( AOwner : TCustomOutline)
17968:  Create( AOwner : TCustomRichEdit)
17969:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
17970:  Create( AOwner : TCustomTreeView)
17971:  Create( AOwner : TIdUserManager)
17972:  Create( AOwner : TListItems)
17973:
       Create(AOwner:TObject;Handl:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvent:TBDECallbckEvent;Chain:Bool)
17974:  CREATE( AOWNER : TPERSISTENT)
17975:  Create( AOwner : TPersistent)
17976:  Create( AOwner : TTable)
17977:  Create( AOwner : TTreeNodes)
17978:  Create( AOwner : TWinControl; const ClassName : string)
17979:  Create( AParent : TIdCustomHTTP)
17980:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
17981:  Create( AProvider : TBaseProvider)
17982:  Create( AProvider : TCustomProvider);
17983:  Create( AProvider : TDataSetProvider)
17984:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
17985:  Create( ASocket : TSocket)
17986:  Create( AStrings : TWideStrings)
17987:  Create( AToolBar : TToolBar)
17988:  Create( ATreeNodes : TTreeNodes)
17989:  Create( Autofill : boolean)
17990:  Create( AWebPageInfo : TAbstractWebPageInfo)
17991:  Create( AWebRequest : TWebRequest)
17992:  Create( Collection : TCollection)
17993:  Create( Collection : TIdMessageParts; ABody : TStrings)
17994:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
17995:  Create( Column : TColumn)
17996:  Create( const AConvFamily : TConvFamily; const ADescription : string)
17997:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
17998:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
      AFromCommonProc : TConversionProc)
17999:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
18000:  Create( const ATabSet : TTabSet)
18001:  Create( const Compensate : Boolean)
18002:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
```

```
18003:  Create( const FileName : string)
18004:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
18005:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
18006:  Create( const MaskValue : string)
18007:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
18008:  Create( const Prefix : string)
18009:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
18010:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18011:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18012:  Create( CoolBar : TCoolBar)
18013:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
18014:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
18015:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
18016:  Create( DBCtrlGrid : TDBCtrlGrid)
18017:  Create( DSTableProducer : TDSTableProducer)
18018:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
18019:  Create( ErrorCode : DBIResult)
18020:  Create( Field : TBlobField; Mode : TBlobStreamMode)
18021:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
18022:  Create( HeaderControl : TCustomHeaderControl)
18023:  Create( HTTPRequest : TWebRequest)
18024:  Create( iStart : integer; sText : string)
18025:  Create( iValue : Integer)
18026:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
18027:  Create( MciErrNo : MCIERROR; const Msg : string)
18028:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
18029:  Create( Message : string; ErrorCode : DBResult)
18030:  Create( Msg : string)
18031:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
18032:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
18033:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
18034:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
18035:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
18036:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
18037:  Create( Owner : TCustomComboBoxEx)
18038:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
18039:  Create( Owner : TPersistent)
18040:  Create( Params : TStrings)
18041:  Create( Size : Cardinal)
18042:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
18043:  Create( StatusBar : TCustomStatusBar)
18044:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
18045:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
18046:  Create(AHandle:Integer)
18047:  Create(AOwner: TComponent); virtual;
18048:  Create(const AURI : string)
18049:  Create(FileName:String;Mode:Word)
18050:  Create(Instance:THandle;ResName:String;ResType:PChar)
18051:  Create(Stream : TStream)
18052:  Create1( ADataset : TDataset);
18053:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
18054:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
18055:  Create2( Other : TObject);
18056:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
18057:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
18058:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
18059:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
18060:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
18061:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
18062:  CreateRes( Ident : Integer);
18063:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
18064:  CreateRes( ResStringRec : PResStringRec);
18065:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
18066:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
18067:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
18068:  CreateSize( AWidth, AHeight : Integer)
18069:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
18070:
18071: ------------------------------------------------------------------------------
18072: unit uPSI_MathMax;
18073: ------------------------------------------------------------------------------
18074:  CONSTS
18075:   Bernstein: Float = 0.28016949902386913303643649123 07;  // Bernstein constant
18076:   Crt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
18077:   Crt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
18078:   Crt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
18079:   Crt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
18080:   CrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
18081:   Catalan: Float  = 0.91596559417721901505460351493 24;  // Catalan constant
18082:   PiJ:   Float      = 3.14159265358979323846264338327 95;  // PI
18083:   PI:  Extended =  3.14159265358979323846264338327 95);
18084:   PiOn2: Float     = 1.57079632679489661923132169163 98;  // PI / 2
18085:   PiOn3: Float     = 1.04719755119659774615421446109 32;  // PI / 3
18086:   PiOn4: Float     = 0.78539816339744830961566084581 988; // PI / 4
18087:   Sqrt2: Float     = 1.41421356237309504880168872420 97;  // Sqrt(2)
```

```
18088:   Sqrt3: Float     = 1.732050807568877293527446341505 9;  // Sqrt(3)
18089:   Sqrt5: Float     = 2.236067977499789696409173668731 3;  // Sqrt(5)
18090:   Sqrt10: Float    = 3.162277660168379331998893544432 7;  // Sqrt(10)
18091:   SqrtPi: Float    = 1.772453850905516027298167483341 1;  // Sqrt(PI)
18092:   Sqrt2Pi: Float   = 2.506628274631000502415765284811;   // Sqrt(2 * PI)
18093:   TwoPi: Float     = 6.283185307179586476925286766559;   // 2 * PI
18094:   ThreePi: Float   = 9.424777960769379715387930149838 5;  // 3 * PI
18095:   Ln2: Float       = 0.693147180559945309417232121458 8; // Ln(2)
18096:   Ln10: Float      = 2.302585092994045684017991454684 4;  // Ln(10)
18097:   LnPi: Float      = 1.144729885849400174143427351353 1;  // Ln(PI)
18098:   Log2J: Float     = 0.301029995663981195213738894724 49; // Log10(2)
18099:   Log3: Float      = 0.477121254719662437295027903255 12; // Log10(3)
18100:   LogPi: Float     = 0.497149872694133854351268288290 9;  // Log10(PI)
18101:   LogE: Float      = 0.434294481903251827651128918916 61; // Log10(E)
18102:   E: Float         = 2.718281828459045235360287471352 7;  // Natural constant
18103:   hLn2Pi: Float    = 0.918938533204672741780329736405 62; // Ln(2*PI)/2
18104:   inv2Pi: Float    = 0.159154943091895335768883763372514362034459645740 46; // 0.5/Pi
18105:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
18106:   GoldenMean: Float   = 1.618033988749894848204586834365638;  // GoldenMean
18107:   EulerMascheroni: Float = 0.577215664901532860606512090082 4;  // Euler GAMMA
18108:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
18109:   StDelta         : Extended = 0.00001;  {delta for difference equations}
18110:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
18111:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
18112:
18113: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
18114: begin
18115:   MetersPerInch = 0.0254; // [1]
18116:   MetersPerFoot = MetersPerInch * 12;
18117:   MetersPerYard = MetersPerFoot * 3;
18118:   MetersPerMile = MetersPerFoot * 5280;
18119:   MetersPerNauticalMiles = 1852;
18120:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
18121:   MetersPerLightSecond = 2.99792458E8; // [5]
18122:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
18123:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
18124:   MetersPerCubit = 0.4572; // [6][7]
18125:   MetersPerFathom = MetersPerFoot * 6;
18126:   MetersPerFurlong = MetersPerYard * 220;
18127:   MetersPerHand = MetersPerInch * 4;
18128:   MetersPerPace = MetersPerInch * 30;
18129:   MetersPerRod = MetersPerFoot * 16.5;
18130:   MetersPerChain = MetersPerRod * 4;
18131:   MetersPerLink = MetersPerChain / 100;
18132:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
18133:   MetersPerPica = MetersPerPoint * 12;
18134:
18135:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
18136:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
18137:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
18138:   SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
18139:   SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
18140:   SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
18141:
18142:   CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
18143:   CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
18144:   CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
18145:   CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
18146:   CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
18147:   CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
18148:   CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
18149:   CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
18150:
18151:   CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
18152:   CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
18153:   CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
18154:   CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
18155:   CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
18156:   CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
18157:   CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
18158:   CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
18159:
18160:   CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
18161:   CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
18162:   CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
18163:   CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
18164:   CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
18165:   CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
18166:
18167:   CubicMetersPerUKGallon = 0.00454609; // [2][7]
18168:   CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
18169:   CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
18170:   CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
18171:   CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
18172:   CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
18173:   CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
18174:   CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
18175:   CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
18176:
```

```
18177:   GramsPerPound = 453.59237; // [1][7]
18178:   GramsPerDrams = GramsPerPound / 256;
18179:   GramsPerGrains = GramsPerPound / 7000;
18180:   GramsPerTons = GramsPerPound * 2000;
18181:   GramsPerLongTons = GramsPerPound * 2240;
18182:   GramsPerOunces = GramsPerPound / 16;
18183:   GramsPerStones = GramsPerPound * 14;
18184:
18185:   MaxAngle 9223372036854775808.0;
18186:   MaxTanH 5678.26170314707197474596553898 54);
18187:   MaxFactorial( 1754);
18188:   MaxFloatingPoint(1.189731495357231765085759326628E+4932);
18189:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
18190:   MaxTanH( 354.89135644669199842162284618659);
18191:   MaxFactorial'LongInt'( 170);
18192:   MaxFloatingPointD(1.797693134862315907729305190789E+308);
18193:   MinFloatingPointD(2.225073858507201383090232717332 4E-308);
18194:   MaxTanH( 44.361419555836499802702855773323);
18195:   MaxFactorial'LongInt'( 33);
18196:   MaxFloatingPointS( 3.402823669209384634633746074317 7E+38);
18197:   MinFloatingPointS( 1.175494350822287507968736537222 2E-38);
18198:   PiExt( 3.14159265358979323846264338327 95);
18199:   RatioDegToRad( PiExt / 180.0);
18200:   RatioGradToRad( PiExt / 200.0);
18201:   RatioDegToGrad( 200.0 / 180.0);
18202:   RatioGradToDeg( 180.0 / 200.0);
18203:   Crc16PolynomCCITT'LongWord $1021);
18204:  Crc16PolynomIBM'LongWord $8005);
18205:  Crc16Bits'LongInt'( 16);
18206:  Crc16Bytes'LongInt'( 2);
18207:  Crc16HighBit'LongWord $8000);
18208:  NotCrc16HighBit','LongWord $7FFF);
18209:   Crc32PolynomIEEE','LongWord $04C11DB7);
18210:  Crc32PolynomCastagnoli','LongWord $1EDC6F41);
18211:  Crc32Koopman','LongWord $741B8CD7);
18212:  Crc32Bits','LongInt'( 32);
18213:  Crc32Bytes','LongInt'( 4);
18214:  Crc32HighBit','LongWord $80000000);
18215:  NotCrc32HighBit','LongWord $7FFFFFFF);
18216:
18217:   MinByte       = Low(Byte);
18218:   MaxByte       = High(Byte);
18219:   MinWord       = Low(Word);
18220:   MaxWord       = High(Word);
18221:   MinShortInt   = Low(ShortInt);
18222:   MaxShortInt   = High(ShortInt);
18223:   MinSmallInt   = Low(SmallInt);
18224:   MaxSmallInt   = High(SmallInt);
18225:   MinLongWord   = LongWord(Low(LongWord));
18226:   MaxLongWord   = LongWord(High(LongWord));
18227:   MinLongInt    = LongInt(Low(LongInt));
18228:   MaxLongInt    = LongInt(High(LongInt));
18229:   MinInt64      = Int64(Low(Int64));
18230:   MaxInt64      = Int64(High(Int64));
18231:   MinInteger    = Integer(Low(Integer));
18232:   MaxInteger    = Integer(High(Integer));
18233:   MinCardinal   = Cardinal(Low(Cardinal));
18234:   MaxCardinal   = Cardinal(High(Cardinal));
18235:   MinNativeUInt = NativeUInt(Low(NativeUInt));
18236:   MaxNativeUInt = NativeUInt(High(NativeUInt));
18237:   MinNativeInt  = NativeInt(Low(NativeInt));
18238:   MaxNativeInt  = NativeInt(High(NativeInt));
18239:  Function CosH( const Z : Float) : Float;
18240:  Function SinH( const Z : Float) : Float;
18241:  Function TanH( const Z : Float) : Float;
18242:
18243:
18244:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
18245:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
18246:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
18247:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
18248:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
18249:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
18250:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
18251:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
18252:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
18253:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
18254:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
18255:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
18256:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
18257:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
18258:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
18259:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
18260:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
18261:   MaxLog = 709.7827128933840;
18262:   MinLog = -708.3964185322641;
18263:   MaxFac = 170;
18264:   MaxGam = 171.624376956302;
18265:   MaxLgm = 2.556348E+305;
```

```
18266:   SingleCompareDelta   = 1.0E-34;
18267:   DoubleCompareDelta   = 1.0E-280;
18268:   {$IFDEF CLR}
18269:   ExtendedCompareDelta = DoubleCompareDelta;
18270:   {$ELSE}
18271:   ExtendedCompareDelta = 1.0E-4400;
18272:   {$ENDIF}
18273:   Bytes1KB  = 1024;
18274:   Bytes1MB  = 1024 * Bytes1KB;
18275:   Bytes1GB  = 1024 * Bytes1MB;
18276:   Bytes64KB = 64 * Bytes1KB;
18277:   Bytes64MB = 64 * Bytes1MB;
18278:   Bytes2GB  = 2 * LongWord(Bytes1GB);
18279:     clBlack32', $FF000000 ));
18280:     clDimGray32', $FF3F3F3F ));
18281:     clGray32', $FF7F7F7F ));
18282:     clLightGray32', $FFBFBFBF ));
18283:     clWhite32', $FFFFFFFF ));
18284:     clMaroon32', $FF7F0000 ));
18285:     clGreen32', $FF007F00 ));
18286:     clOlive32', $FF7F7F00 ));
18287:     clNavy32', $FF00007F ));
18288:     clPurple32', $FF7F007F ));
18289:     clTeal32', $FF007F7F ));
18290:     clRed32', $FFFF0000 ));
18291:     clLime32', $FF00FF00 ));
18292:     clYellow32', $FFFFFF00 ));
18293:     clBlue32', $FF0000FF ));
18294:     clFuchsia32', $FFFF00FF ));
18295:     clAqua32', $FF00FFFF ));
18296:     clAliceBlue32', $FFF0F8FF ));
18297:     clAntiqueWhite32', $FFFAEBD7 ));
18298:     clAquamarine32', $FF7FFFD4 ));
18299:     clAzure32', $FFF0FFFF ));
18300:     clBeige32', $FFF5F5DC ));
18301:     clBisque32', $FFFFE4C4 ));
18302:     clBlancheDalmond32', $FFFFEBCD ));
18303:     clBlueViolet32', $FF8A2BE2 ));
18304:     clBrown32', $FFA52A2A ));
18305:     clBurlyWood32', $FFDEB887 ));
18306:     clCadetblue32', $FF5F9EA0 ));
18307:     clChartReuse32', $FF7FFF00 ));
18308:     clChocolate32', $FFD2691E ));
18309:     clCoral32', $FFFF7F50 ));
18310:     clCornFlowerBlue32', $FF6495ED ));
18311:     clCornSilk32', $FFFFF8DC ));
18312:     clCrimson32', $FFDC143C ));
18313:     clDarkBlue32', $FF00008B ));
18314:     clDarkCyan32', $FF008B8B ));
18315:     clDarkGoldenRod32', $FFB8860B ));
18316:     clDarkGray32', $FFA9A9A9 ));
18317:     clDarkGreen32', $FF006400 ));
18318:     clDarkGrey32', $FFA9A9A9 ));
18319:     clDarkKhaki32', $FFBDB76B ));
18320:     clDarkMagenta32', $FF8B008B ));
18321:     clDarkOliveGreen32', $FF556B2F ));
18322:     clDarkOrange32', $FFFF8C00 ));
18323:     clDarkOrchid32', $FF9932CC ));
18324:     clDarkRed32', $FF8B0000 ));
18325:     clDarkSalmon32', $FFE9967A ));
18326:     clDarkSeaGreen32', $FF8FBC8F ));
18327:     clDarkSlateBlue32', $FF483D8B ));
18328:     clDarkSlateGray32', $FF2F4F4F ));
18329:     clDarkSlateGrey32', $FF2F4F4F ));
18330:     clDarkTurquoise32', $FF00CED1 ));
18331:     clDarkViolet32', $FF9400D3 ));
18332:     clDeepPink32', $FFFF1493 ));
18333:     clDeepSkyBlue32', $FF00BFFF ));
18334:     clDodgerBlue32', $FF1E90FF ));
18335:     clFireBrick32', $FFB22222 ));
18336:     clFloralWhite32', $FFFFFAF0 ));
18337:     clGainsBoro32', $FFDCDCDC ));
18338:     clGhostWhite32', $FFF8F8FF ));
18339:     clGold32', $FFFFD700 ));
18340:     clGoldenRod32', $FFDAA520 ));
18341:     clGreenYellow32', $FFADFF2F ));
18342:     clGrey32', $FF808080 ));
18343:     clHoneyDew32', $FFF0FFF0 ));
18344:     clHotPink32', $FFFF69B4 ));
18345:     clIndianRed32', $FFCD5C5C ));
18346:     clIndigo32', $FF4B0082 ));
18347:     clIvory32', $FFFFFFF0 ));
18348:     clKhaki32', $FFF0E68C ));
18349:     clLavender32', $FFE6E6FA ));
18350:     clLavenderBlush32', $FFFFF0F5 ));
18351:     clLawnGreen32', $FF7CFC00 ));
18352:     clLemonChiffon32', $FFFFFACD ));
18353:     clLightBlue32', $FFADD8E6 ));
18354:     clLightCoral32', $FFF08080 ));
```

```
18355:     clLightCyan32', $FFE0FFFF ));
18356:     clLightGoldenRodYellow32', $FFFAFAD2 ));
18357:     clLightGreen32', $FF90EE90 ));
18358:     clLightGrey32', $FFD3D3D3 ));
18359:     clLightPink32', $FFFFB6C1 ));
18360:     clLightSalmon32', $FFFFA07A ));
18361:     clLightSeagreen32', $FF20B2AA ));
18362:     clLightSkyblue32', $FF87CEFA ));
18363:     clLightSlategray32', $FF778899 ));
18364:     clLightSlategrey32', $FF778899 ));
18365:     clLightSteelblue32', $FFB0C4DE ));
18366:     clLightYellow32', $FFFFFFE0 ));
18367:     clLtGray32', $FFC0C0C0 ));
18368:     clMedGray32', $FFA0A0A4 ));
18369:     clDkGray32', $FF808080 ));
18370:     clMoneyGreen32', $FFC0DCC0 ));
18371:     clLegacySkyBlue32', $FFA6CAF0 ));
18372:     clCream32', $FFFFFBF0 ));
18373:     clLimeGreen32', $FF32CD32 ));
18374:     clLinen32', $FFFAF0E6 ));
18375:     clMediumAquamarine32', $FF66CDAA ));
18376:     clMediumBlue32', $FF0000CD ));
18377:     clMediumOrchid32', $FFBA55D3 ));
18378:     clMediumPurple32', $FF9370DB ));
18379:     clMediumSeaGreen32', $FF3CB371 ));
18380:     clMediumSlateBlue32', $FF7B68EE ));
18381:     clMediumSpringGreen32', $FF00FA9A ));
18382:     clMediumTurquoise32', $FF48D1CC ));
18383:     clMediumVioletRed32', $FFC71585 ));
18384:     clMidnightBlue32', $FF191970 ));
18385:     clMintCream32', $FFF5FFFA ));
18386:     clMistyRose32', $FFFFE4E1 ));
18387:     clMoccasin32', $FFFFE4B5 ));
18388:     clNavajoWhite32', $FFFFDEAD ));
18389:     clOldLace32', $FFFDF5E6 ));
18390:     clOliveDrab32', $FF6B8E23 ));
18391:     clOrange32', $FFFFA500 ));
18392:     clOrangeRed32', $FFFF4500 ));
18393:     clOrchid32', $FFDA70D6 ));
18394:     clPaleGoldenRod32', $FFEEE8AA ));
18395:     clPaleGreen32', $FF98FB98 ));
18396:     clPaleTurquoise32', $FFAFEEEE ));
18397:     clPaleVioletred32', $FFDB7093 ));
18398:     clPapayaWhip32', $FFFFEFD5 ));
18399:     clPeachPuff32', $FFFFDAB9 ));
18400:     clPeru32', $FFCD853F ));
18401:     clPlum32', $FFDDA0DD ));
18402:     clPowderBlue32', $FFB0E0E6 ));
18403:     clRosyBrown32', $FFBC8F8F ));
18404:     clRoyalBlue32', $FF4169E1 ));
18405:     clSaddleBrown32', $FF8B4513 ));
18406:     clSalmon32', $FFFA8072 ));
18407:     clSandyBrown32', $FFF4A460 ));
18408:     clSeaGreen32', $FF2E8B57 ));
18409:     clSeaShell32', $FFFFF5EE ));
18410:     clSienna32', $FFA0522D ));
18411:     clSilver32', $FFC0C0C0 ));
18412:     clSkyblue32', $FF87CEEB ));
18413:     clSlateBlue32', $FF6A5ACD ));
18414:     clSlateGray32', $FF708090 ));
18415:     clSlateGrey32', $FF708090 ));
18416:     clSnow32', $FFFFFAFA ));
18417:     clSpringgreen32', $FF00FF7F ));
18418:     clSteelblue32', $FF4682B4 ));
18419:     clTan32', $FFD2B48C ));
18420:     clThistle32', $FFD8BFD8 ));
18421:     clTomato32', $FFFF6347 ));
18422:     clTurquoise32', $FF40E0D0 ));
18423:     clViolet32', $FFEE82EE ));
18424:     clWheat32', $FFF5DEB3 ));
18425:     clWhitesmoke32', $FFF5F5F5 ));
18426:     clYellowgreen32', $FF9ACD32 ));
18427:     clTrWhite32', $7FFFFFFF ));
18428:     clTrBlack32', $7F000000 ));
18429:     clTrRed32', $7FFF0000 ));
18430:     clTrGreen32', $7F00FF00 ));
18431:     clTrBlue32', $7F0000FF ));
18432:   // Fixed point math constants
18433:   FixedOne = $10000;  FixedHalf = $7FFF;
18434:   FixedPI  = Round(PI * FixedOne);
18435:   FixedToFloat = 1/FixedOne;
18436:
18437:  Special Types
18438: ************************************************
18439:   type Complex = record
18440:     X, Y : Float;
18441:   end;
18442:   type TVector     = array of Float;
18443:   TIntVector  = array of Integer;
```

```
18444:   TCompVector = array of Complex;
18445:   TBoolVector = array of Boolean;
18446:   TStrVector  = array of String;
18447:   TMatrix     = array of TVector;
18448:   TIntMatrix  = array of TIntVector;
18449:   TCompMatrix = array of TCompVector;
18450:   TBoolMatrix = array of TBoolVector;
18451:   TStrMatrix  = array of TStrVector;
18452:   TByteArray = array[0..32767] of byte; !
18453:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
18454:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
18455:   T2StringArray = array of array of string;
18456:   T2IntegerArray = array of array of integer;
18457:   AddTypeS('INT_PTR', 'Integer
18458:   AddTypeS('LONG_PTR', 'Integer
18459:   AddTypeS('UINT_PTR', 'Cardinal
18460:   AddTypeS('ULONG_PTR', 'Cardinal
18461:   AddTypeS('DWORD_PTR', 'ULONG_PTR
18462:   TIntegerDynArray', 'array of Integer
18463:   TCardinalDynArray', 'array of Cardinal
18464:   TWordDynArray', 'array of Word
18465:   TSmallIntDynArray', 'array of SmallInt
18466:   TByteDynArray', 'array of Byte
18467:   TShortIntDynArray', 'array of ShortInt
18468:   TInt64DynArray', 'array of Int64
18469:   TLongWordDynArray', 'array of LongWord
18470:   TSingleDynArray', 'array of Single
18471:   TDoubleDynArray', 'array of Double
18472:   TBooleanDynArray', 'array of Boolean
18473:   TStringDynArray', 'array of string
18474:   TWideStringDynArray', 'array of WideString
18475:   TDynByteArray     = array of Byte;
18476:   TDynShortintArray = array of Shortint;
18477:   TDynSmallintArray = array of Smallint;
18478:   TDynWordArray     = array of Word;
18479:   TDynIntegerArray  = array of Integer;
18480:   TDynLongintArray  = array of Longint;
18481:   TDynCardinalArray = array of Cardinal;
18482:   TDynInt64Array    = array of Int64;
18483:   TDynExtendedArray = array of Extended;
18484:   TDynDoubleArray   = array of Double;
18485:   TDynSingleArray   = array of Single;
18486:   TDynFloatArray    = array of Float;
18487:   TDynPointerArray  = array of Pointer;
18488:   TDynStringArray   = array of string;
18489:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
18490:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
18491:   TSynSearchOptions = set of TSynSearchOption;
18492:
18493:
18494:
18495: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
18496: ----------------------------------------------------------------------------
18497: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
18498: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
18499: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
18500: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
18501: function CheckStringSum(vstring: string): integer;
18502: function HexToInt(HexNum: string): LongInt;
18503: function IntToBin(Int: Integer): String;
18504: function BinToInt(Binary: String): Integer;
18505: function HexToBin(HexNum: string): string; external2
18506: function BinToHex(Binary: String): string;
18507: function IntToFloat(i: Integer): double;
18508: function AddThousandSeparator(S: string; myChr: Char): string;
18509: function Max3(const X,Y,Z: Integer): Integer;
18510: procedure Swap(var X,Y: char); // faster without inline
18511: procedure ReverseString(var S: String);
18512: function CharToHexStr(Value: Char): string;
18513: function CharToUniCode(Value: Char): string;
18514: function Hex2Dec(Value: Str002): Byte;
18515: function HexStrCodeToStr(Value: string): string;
18516: function HexToStr(i: integer; value: string): string;
18517: function UniCodeToStr(Value: string): string;
18518: function CRC16(statement: string): string;
18519: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
18520: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
18521: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
18522: Procedure ExecuteCommand(executeFile, paramstring: string);
18523: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
18524: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
18525: procedure SearchAndOpenDoc(vfilenamepath: string);
18526: procedure ShowInterfaces(myFile: string);
18527: function Fact2(av: integer): extended;
18528: Function BoolToStr(B: Boolean): string;
18529: Function GCD(x, y : LongInt) : LongInt;
18530: function LCM(m,n: longint): longint;
18531: function GetASCII: string;
18532: function GetItemHeight(Font: TFont): Integer;
```

```
18533: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
18534: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
18535: function getHINSTANCE: longword;
18536: function getHMODULE: longword;
18537: function GetASCII: string;
18538: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
18539: function WordIsOk(const AWord: string; var VW: Word): boolean;
18540: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
18541: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
18542: function SafeStr(const s: string): string;
18543: function ExtractUrlPath(const FileName: string): string;
18544: function ExtractUrlName(const FileName: string): string;
18545: function IsInternet: boolean;
18546: function RotateLeft1Bit_u32( Value: uint32): uint32;
18547: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats : Boolean);
18548: procedure getEnvironmentInfo;
18549: procedure AntiFreeze;
18550: function GetCPUSpeed: Double;
18551: function IsVirtualPcGuest : Boolean;
18552: function IsVmWareGuest : Boolean;
18553: procedure StartSerialDialog;
18554: function IsWoW64: boolean;
18555: function IsWow64String(var s: string): Boolean;
18556: procedure StartThreadDemo;
18557: Function RGB(R,G,B: Byte): TColor;
18558: Function Sendln(amess: string): boolean;
18559: Procedure maXbox;
18560: Function AspectRatio(aWidth, aHeight: Integer): String;
18561: function wget(aURL, afile: string): boolean;
18562: procedure PrintList(Value: TStringList);
18563: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
18564: procedure getEnvironmentInfo;
18565: procedure AntiFreeze;
18566: function getBitmap(apath: string): TBitmap;
18567: procedure ShowMessageBig(const aText : string);
18568: function YesNoDialog(const ACaption, AMsg: string): boolean;
18569: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
18570: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
18571: //function myStrToBytes(const Value: String): TBytes;
18572: //function myBytesToStr(const Value: TBytes): String;
18573: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
18574: function getBitmap(apath: string): TBitmap;
18575: procedure ShowMessageBig(const aText : string);
18576: Function StrToBytes(const Value: String): TBytes;
18577: Function BytesToStr(const Value: TBytes): String;
18578: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
18579: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retris:Int;var
       HostName:String):Bool;
18580: function FindInPaths(const fileName, paths : String) : String;
18581: procedure initHexArray(var hexn: THexArray);
18582: function josephusG(n,k: integer; var graphout: string): integer;
18583: function isPowerof2(num: int64): boolean;
18584: function powerOf2(exponent: integer): int64;
18585: function getBigPI: string;
18586: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
18587:  function GetASCIILine: string;
18588: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
18589:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
18590: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
18591: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
18592: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
18593: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
18594: function isKeypressed: boolean;
18595: function Keypress: boolean;
18596: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
18597: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
18598: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
18599: function GetOSName: string;
18600: function GetOSVersion: string;
18601: function GetOSNumber: string;
18602: function getEnvironmentString: string;
18603: procedure StrReplace(var Str: String; Old, New: String);
18604: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
18605: function getTeamViewerID: string;
18606: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
18607: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
18608: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
18609: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
18610: function StartSocketService: Boolean;
18611: procedure StartSocketServiceForm;
18612: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
18613: function GetFileList1(apath: string): TStringlist;
18614: procedure LetFileList(FileList: TStringlist; apath: string);
18615: procedure StartWeb(aurl: string);
18616: function GetTodayFiles(startdir, amask: string): TStringlist;
18617: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
18618: function JavahashCode(val: string): Integer;
18619: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
```

```
18620: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
18621: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
18622: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
18623: Procedure ConvertToGray(Cnv: TCanvas);
18624: function GetFileDate(aFile:string; aWithTime:Boolean):string;
18625: procedure ShowMemory;
18626: function ShowMemory2: string;
18627: function getHostIP: string;
18628: procedure ShowBitmap(bmap: TBitmap);
18629: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
18630: function CreateDBGridForm(dblist: TStringList): TListbox;
18631: function isService: boolean;
18632: function isApplication: boolean;
18633: function isTerminalSession: boolean;
18634:
18635:
18636: // News of 3.9.8 up
18637: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
18638: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
18639: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
18640: JvChart - TJvChart Component - 2009 Public
18641: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
18642: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
18643: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
18644: DMath DLL included incl. Demos
18645: Interface Navigator menu/View/Intf Navigator
18646: Unit Explorer menu/Debug/Units Explorer
18647: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
18648: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
18649: Script History to 9 Files WebServer light /Options/Addons/WebServer
18650: Full Text Finder, JVSimLogic Simulator Package
18651: Halt-Stop Program in Menu, WebServer2, Stop Event ,
18652: Conversion Routines, Prebuild Forms, CodeSearch
18653: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
18654: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
18655: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
18656: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
18657: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
18658: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
18659: IDE Reflection API, Session Service Shell S3
18660: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
18661: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
18662: arduino map() function, PMRandom Generator
18663: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
18664: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
18665: REST Test Lib, Multilang Component, Forth Interpreter
18666: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
18667: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
18668: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
18669: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
18670: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
18671: QRCode Service, add more CFunctions like CDateTime of Synapse
18672: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
18673: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
18674: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
18675: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
18676: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
18677: BOLD Package, Indy Package5, maTRIx. MATHEMAX
18678: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
18679: emax layers: system-package-component-unit-class-function-block
18680: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
18681: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
18682: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
18683: OpenGL Game Demo: ..Options/Add Ons/Reversi
18684: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
18685: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
18686: 7% performance gain (hot spot profiling)
18687: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
18688: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
18689: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
18690: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
18691:
18692: add routines in 3.9.7.5
18693: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
18694: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
18695: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
18696: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
18697: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
18698: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
18699: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
18700:
18701: ////////////////////////// TestUnits ////////////////////////////
18702:   SelftestPEM;
18703:   SelfTestCFundamentUtils;
18704:   SelfTestCFileUtils;
18705:   SelfTestCDateTime;
18706:   SelfTestCTimer;
18707:   SelfTestCRandom;
18708:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
```

```
18709:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
18710:
18711:   // Note: There's no need for installing a client certificate in the
18712:   //       webbrowser. The server asks the webbrowser to send a certificate but
18713:   //       if nothing is installed the software will work because the server
18714:   //       doesn't check to see if a client certificate was supplied. If you want you can install:
18715:   //
18716:   //       file: c_cacert.p12
18717:   //       password: c_cakey
18718:
18719:   TGraphicControl = class(TControl)
18720:   private
18721:     FCanvas: TCanvas;
18722:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
18723:   protected
18724:     procedure Paint; virtual;
18725:     property Canvas: TCanvas read FCanvas;
18726:   public
18727:     constructor Create(AOwner: TComponent); override;
18728:     destructor Destroy; override;
18729:   end;
18730:
18731:   TCustomControl = class(TWinControl)
18732:   private
18733:     FCanvas: TCanvas;
18734:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
18735:   protected
18736:     procedure Paint; virtual;
18737:     procedure PaintWindow(DC: HDC); override;
18738:     property Canvas: TCanvas read FCanvas;
18739:   public
18740:     constructor Create(AOwner: TComponent); override;
18741:     destructor Destroy; override;
18742:   end;
18743:     RegisterPublishedProperties;
18744:     ('ONCHANGE', 'TNotifyEvent', iptrw);
18745:     ('ONCLICK', 'TNotifyEvent', iptrw);
18746:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
18747:     ('ONENTER', 'TNotifyEvent', iptrw);
18748:     ('ONEXIT', 'TNotifyEvent', iptrw);
18749:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
18750:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
18751:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
18752:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
18753:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
18754: //*********************************************************************
18755:   // To stop the while loop, click on Options/Show Include (boolean switch)!
18756:   Control a loop in a script with a form event:
18757:   IncludeON;   //control the while loop
18758:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
18759:
18760: //---------------------------------------------------------------------------
18761: //*************mX4 ini-file Configuration**********************************
18762: //---------------------------------------------------------------------------
18763:   using config file maxboxdef.ini       menu/Help/Config File
18764:
18765: //*** Definitions for maXbox mX3 ***
18766: [FORM]
18767: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
18768: FONTSIZE=14
18769: EXTENSION=txt
18770: SCREENX=1386
18771: SCREENY=1077
18772: MEMHEIGHT=350
18773: PRINTFONT=Courier New //GUI Settings
18774: LINENUMBERS=Y   //line numbers at gutter in editor at left side
18775: EXCEPTIONLOG=Y  //store excepts and success in 2 log files see below! –menu Debug/Show Last Exceptions
18776: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
18777: BOOTSCRIPT=Y    //enabling load a boot script
18778: MEMORYREPORT=Y  //shows memory report on closing maXbox
18779: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
18780: NAVIGATOR=N     //shows function list at the right side of editor
18781: NAVWIDTH=350    //width of the right side interface list <CTRL L>  >=200
18782: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
18783: [WEB]
18784: IPPORT=8080     //for internal webserver – menu /Options/Add Ons/WebServer
18785: IPHOST=192.168.1.53
18786: ROOTCERT=filepathY
18787: SCERT=filepathY
18788: RSAKEY=filepathY
18789: VERSIONCHECK=Y
18790:
18791: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
18792:
18793: Also possible to set report memory in script to override ini setting
18794: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
18795:
18796:
18797: After Change the ini file you can reload the file with ../Help/Config Update
```

```
18798:
18799: //-------------------------------------------------------------------------
18800: //**************mX4 maildef.ini ini-file Configuration********************
18801: //-------------------------------------------------------------------------
18802: //*** Definitions for maXMail ***
18803: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
18804: [MAXMAIL]
18805: HOST=getmail.softwareschule.ch
18806: USER=mailusername
18807: PASS=password
18808: PORT=110
18809: SSL=Y
18810: BODY=Y
18811: LAST=5
18812:
18813: ADO Connection String:
18814: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
18815:
18816: OpenSSL Lib:  unit ssl_openssl_lib;
18817: {$IFDEF CIL}
18818: const
18819:   {$IFDEF LINUX}
18820:   DLLSSLName = 'libssl.so';
18821:   DLLUtilName = 'libcrypto.so';
18822:   {$ELSE}
18823:   DLLSSLName = 'ssleay32.dll';
18824:   DLLUtilName = 'libeay32.dll';
18825:   {$ENDIF}
18826: {$ELSE}
18827: var
18828:   {$IFNDEF MSWINDOWS}
18829:     {$IFDEF DARWIN}
18830:     DLLSSLName: string = 'libssl.dylib';
18831:     DLLUtilName: string = 'libcrypto.dylib';
18832:     {$ELSE}
18833:     DLLSSLName: string = 'libssl.so';
18834:     DLLUtilName: string = 'libcrypto.so';
18835:     {$ENDIF}
18836:   {$ELSE}
18837:   DLLSSLName: string = 'ssleay32.dll';
18838:   DLLSSLName2: string = 'libssl32.dll';
18839:   DLLUtilName: string = 'libeay32.dll';
18840:   {$ENDIF}
18841: {$ENDIF}
18842:
18843:
18844: //-------------------------------------------------------------------------
18845: //**************mX4 Macro Tags  ********************************************
18846: //-------------------------------------------------------- ----------------
18847:
18848:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
     E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
18849:
18850: //Tag Macros
18851:
18852:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
18853:
18854: //Tag Macros
18855: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
18856: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
18857: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
18858: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
18859: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
18860: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
18861: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
18862: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
18863: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
18864:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
18865: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
18866: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
18867:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
18868: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
18869:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
18870: 10298:  SearchAndCopy(memo1.lines, '#net',Format('DNS: %s; local IPs: %s; local IP: %s',
18871:        [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)]), 10);
18872:
18873: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
18874:
18875: //Replace Macros
18876:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
18877:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
18878:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
18879:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
18880:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
18881:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
18882:
18883:   SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
18884:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
18885: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
```

```
18886:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
18887:
18888: //------------------------------------------------------------------------------
18889: //**************mX4 ToDo List Tags  ../Help/ToDo List************************
18890: //------------------------------------------------------------ ---------------------
18891:
18892:     while I < sl.Count do begin
18893: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
18894:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
18895:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
18896:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
18897:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
18898:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
18899:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
18900:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
18901:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
18902:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
18903:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
18904:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
18905:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
18906:       Inc(I);
18907:     end;
18908:
18909:
18910: //------------------------------------------------------------------------------
18911: //**************mX4 Public  Tools API ***************************************
18912: //------------------------------------------------------------------------------
18913:   file : unit uPSI_fMain.pas;              {$OTAP} Open Tools API Catalog
18914:  // Those functions concern the editor and preprocessor, all of the IDE
18915:  Example: Call it with maxform1.Info1Click(self)
18916:  Note: Call all Methods with maxForm1., e.g.:
18917:                       maxForm1.ShellStyle1Click(self);
18918:
18919: procedure SIRegister_fMain(CL: TPSPascalCompiler);
18920: begin
18921:  Const('BYTECODE','String 'bytecode.txt'
18922:  Const('PSTEXT','String PS Scriptfiles (*.txt)|*.TXT)'
18923:  Const('PSMODEL','String PS Modelfiles (*.uc)|*.UC
18924:  Const('PSPASCAL','String PS Pascalfiles (*.pas)|*.PAS
18925:  Const('PSINC','String PS Includes (*.inc)|*.INC
18926:  Const('DEFFILENAME','String 'firstdemo.txt
18927:  Const('DEFINIFILE','String 'maxboxdef.ini
18928:  Const('EXCEPTLOGFILE','String 'maxboxerrorlog.txt
18929:  Const('ALLFUNCTIONSLIST','String 'upsi_allfunctionslist.txt
18930:  Const('ALLFUNCTIONSLISTPDF','String 'maxbox_functions_all.pdf
18931:  Const('ALLOBJECTSLIST','String 'docs\VCL.pdf
18932:  Const('ALLRESOURCELIST','String 'docs\upsi_allresourcelist.txt
18933:  Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
18934:  Const('INCLUDEBOX','String 'pas_includebox.inc
18935:  Const('BOOTSCRIPT','String 'maxbootscript.txt
18936:  Const('MBVERSION','String '3.9.9.95
18937:  Const('VERSION','String'3.9.9.95
18938:  Const('MBVER','String '399
18939:  Const('MBVERI','Integer'(399);
18940:  Const('MBVERIALL','Integer'(39995);
18941:  Const('EXENAME','String 'maXbox3.exe
18942:  Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
18943:  Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
18944:  Const('MXINTERNETCHECK','String 'www.ask.com
18945:  Const('MXMAIL','String 'max@kleiner.com
18946:  Const('TAB','Char #$09);
18947:  Const('CODECOMPLETION','String 'bds_delphi.dci
18948:  SIRegister_TMaxForm1(CL);
18949: end;
18950:
18951:   with FindClass('TForm'),'TMaxForm1') do begin
18952:     memo2', 'TMemo', iptrw);
18953:     memo1', 'TSynMemo', iptrw);
18954:     CB1SCList', 'TComboBox', iptrw);
18955:     mxNavigator', 'TComboBox', iptrw);
18956:     IPHost', 'string', iptrw);
18957:     IPPort', 'integer', iptrw);
18958:     COMPort', 'integer', iptrw);      //3.9.6.4
18959:     Splitter1', 'TSplitter', iptrw);
18960:     PSScript', 'TPSScript', iptrw);
18961:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
18962:     MainMenu1', 'TMainMenu', iptrw);
18963:     Program1', 'TMenuItem', iptrw);
18964:     Compile1', 'TMenuItem', iptrw);
18965:     Files1', 'TMenuItem', iptrw);
18966:     open1', 'TMenuItem', iptrw);
18967:     Save2', 'TMenuItem', iptrw);
18968:     Options1', 'TMenuItem', iptrw);
18969:     Savebefore1', 'TMenuItem', iptrw);
18970:     Largefont1', 'TMenuItem', iptrw);
18971:     sBytecode1', 'TMenuItem', iptrw);
18972:     Saveas3', 'TMenuItem', iptrw);
18973:     Clear1', 'TMenuItem', iptrw);
18974:     Slinenumbers1', 'TMenuItem', iptrw);
```

```
18975:     About1', 'TMenuItem', iptrw);
18976:     Search1', 'TMenuItem', iptrw);
18977:     SynPasSyn1', 'TSynPasSyn', iptrw);
18978:     memo1', 'TSynMemo', iptrw);
18979:     SynEditSearch1', 'TSynEditSearch', iptrw);
18980:     WordWrap1', 'TMenuItem', iptrw);
18981:     XPManifest1', 'TXPManifest', iptrw);
18982:     SearchNext1', 'TMenuItem', iptrw);
18983:     Replace1', 'TMenuItem', iptrw);
18984:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
18985:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
18986:     ShowInclude1', 'TMenuItem', iptrw);
18987:     SynEditPrint1', 'TSynEditPrint', iptrw);
18988:     Printout1', 'TMenuItem', iptrw);
18989:     mnPrintColors1', 'TMenuItem', iptrw);
18990:     dlgFilePrint', 'TPrintDialog', iptrw);
18991:     dlgPrintFont1', 'TFontDialog', iptrw);
18992:     mnuPrintFont1', 'TMenuItem', iptrw);
18993:     Include1', 'TMenuItem', iptrw);
18994:     CodeCompletionList1', 'TMenuItem', iptrw);
18995:     IncludeList1', 'TMenuItem', iptrw);
18996:     ImageList1', 'TImageList', iptrw);
18997:     ImageList2', 'TImageList', iptrw);
18998:     CoolBar1', 'TCoolBar', iptrw);
18999:     ToolBar1', 'TToolBar', iptrw);
19000:     tbtnLoad', 'TToolButton', iptrw);
19001:     ToolButton2', 'TToolButton', iptrw);
19002:     tbtnFind', 'TToolButton', iptrw);
19003:     tbtnCompile', 'TToolButton', iptrw);
19004:     tbtnTrans', 'TToolButton', iptrw);
19005:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
19006:     toolbtnTutorial', 'TToolButton', iptrw);
19007:     tbtn6res', 'TToolButton', iptrw);
19008:     ToolButton5', 'TToolButton', iptrw);
19009:     ToolButton1', 'TToolButton', iptrw);
19010:     ToolButton3', 'TToolButton', iptrw);
19011:     statusBar1', 'TStatusBar', iptrw);
19012:     SaveOutput1', 'TMenuItem', iptrw);
19013:     ExportClipboard1', 'TMenuItem', iptrw);
19014:     Close1', 'TMenuItem', iptrw);
19015:     Manual1', 'TMenuItem', iptrw);
19016:     About2', 'TMenuItem', iptrw);
19017:     loadLastfile1', 'TMenuItem', iptrw);
19018:     imglogo', 'TImage', iptrw);
19019:     cedebug', 'TPSScriptDebugger', iptrw);
19020:     debugPopupMenu1', 'TPopupMenu', iptrw);
19021:     BreakPointMenu', 'TMenuItem', iptrw);
19022:     Decompile1', 'TMenuItem', iptrw);
19023:     StepInto1', 'TMenuItem', iptrw);
19024:     StepOut1', 'TMenuItem', iptrw);
19025:     Reset1', 'TMenuItem', iptrw);
19026:     DebugRun1', 'TMenuItem', iptrw);
19027:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
19028:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
19029:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
19030:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
19031:     tutorial4', 'TMenuItem', iptrw);
19032:     ExporttoClipboard1', 'TMenuItem', iptrw);
19033:     ImportfromClipboard1', 'TMenuItem', iptrw);
19034:     N4', 'TMenuItem', iptrw);
19035:     N5', 'TMenuItem', iptrw);
19036:     N6', 'TMenuItem', iptrw);
19037:     ImportfromClipboard2', 'TMenuItem', iptrw);
19038:     tutorial1', 'TMenuItem', iptrw);
19039:     N7', 'TMenuItem', iptrw);
19040:     ShowSpecChars1', 'TMenuItem', iptrw);
19041:     OpenDirectory1', 'TMenuItem', iptrw);
19042:     procMess', 'TMenuItem', iptrw);
19043:     tbtnUseCase', 'TToolButton', iptrw);
19044:     ToolButton7', 'TToolButton', iptrw);
19045:     EditFont1', 'TMenuItem', iptrw);
19046:     UseCase1', 'TMenuItem', iptrw);
19047:     tutorial21', 'TMenuItem', iptrw);
19048:     OpenUseCase1', 'TMenuItem', iptrw);
19049:     PSImport_DB1', 'TPSImport_DB', iptrw);
19050:     tutorial31', 'TMenuItem', iptrw);
19051:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
19052:     HTMLSyntax1', 'TMenuItem', iptrw);
19053:     ShowInterfaces1', 'TMenuItem', iptrw);
19054:     Tutorial5', 'TMenuItem', iptrw);
19055:     AllFunctionsList1', 'TMenuItem', iptrw);
19056:     ShowLastException1', 'TMenuItem', iptrw);
19057:     PlayMP31', 'TMenuItem', iptrw);
19058:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
19059:     texSyntax1', 'TMenuItem', iptrw);
19060:     N8', 'TMenuItem', iptrw);
19061:     GetEMails1', 'TMenuItem', iptrw);
19062:     SynCppSyn1', 'TSynCppSyn', iptrw);
19063:     CSyntax1', 'TMenuItem', iptrw);
```

```
19064:     Tutorial6', 'TMenuItem', iptrw);
19065:     New1', 'TMenuItem', iptrw);
19066:     AllObjectsList1', 'TMenuItem', iptrw);
19067:     LoadBytecode1', 'TMenuItem', iptrw);
19068:     CipherFile1', 'TMenuItem', iptrw);
19069:     N9', 'TMenuItem', iptrw);
19070:     N10', 'TMenuItem', iptrw);
19071:     Tutorial11', 'TMenuItem', iptrw);
19072:     Tutorial71', 'TMenuItem', iptrw);
19073:     UpdateService1', 'TMenuItem', iptrw);
19074:     PascalSchool1', 'TMenuItem', iptrw);
19075:     Tutorial81', 'TMenuItem', iptrw);
19076:     DelphiSite1', 'TMenuItem', iptrw);
19077:     Output1', 'TMenuItem', iptrw);
19078:     TerminalStyle1', 'TMenuItem', iptrw);
19079:     ReadOnly1', 'TMenuItem', iptrw);
19080:     ShellStyle1', 'TMenuItem', iptrw);
19081:     BigScreen1', 'TMenuItem', iptrw);
19082:     Tutorial91', 'TMenuItem', iptrw);
19083:     SaveOutput2', 'TMenuItem', iptrw);
19084:     N11', 'TMenuItem', iptrw);
19085:     SaveScreenshot', 'TMenuItem', iptrw);
19086:     Tutorial101', 'TMenuItem', iptrw);
19087:     SQLSyntax1', 'TMenuItem', iptrw);
19088:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
19089:     Console1', 'TMenuItem', iptrw);
19090:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
19091:     XMLSyntax1', 'TMenuItem', iptrw);
19092:     ComponentCount1', 'TMenuItem', iptrw);
19093:     NewInstance1', 'TMenuItem', iptrw);
19094:     toolbtnTutorial', 'TToolButton', iptrw);
19095:     Memory1', 'TMenuItem', iptrw);
19096:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
19097:     JavaSyntax1', 'TMenuItem', iptrw);
19098:     SyntaxCheck1', 'TMenuItem', iptrw);
19099:     Tutorial10Statistics1', 'TMenuItem', iptrw);
19100:     ScriptExplorer1', 'TMenuItem', iptrw);
19101:     FormOutput1', 'TMenuItem', iptrw);
19102:     ArduinoDump1', 'TMenuItem', iptrw);
19103:     AndroidDump1', 'TMenuItem', iptrw);
19104:     GotoEnd1', 'TMenuItem', iptrw);
19105:     AllResourceList1', 'TMenuItem', iptrw);
19106:     ToolButton4', 'TToolButton', iptrw);
19107:     tbtn6res', 'TToolButton', iptrw);
19108:     Tutorial11Forms1', 'TMenuItem', iptrw);
19109:     Tutorial12SQL1', 'TMenuItem', iptrw);
19110:     ResourceExplore1', 'TMenuItem', iptrw);
19111:     Info1', 'TMenuItem', iptrw);
19112:     N12', 'TMenuItem', iptrw);
19113:     CryptoBox1', 'TMenuItem', iptrw);
19114:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
19115:     CipherFile2', 'TMenuItem', iptrw);
19116:     N13', 'TMenuItem', iptrw);
19117:     ModulesCount1', 'TMenuItem', iptrw);
19118:     AddOns2', 'TMenuItem', iptrw);
19119:     N4GewinntGame1', 'TMenuItem', iptrw);
19120:     DocuforAddOns1', 'TMenuItem', iptrw);
19121:     Tutorial14Async1', 'TMenuItem', iptrw);
19122:     Lessons15Review1', 'TMenuItem', iptrw);
19123:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
19124:     PHPSyntax1', 'TMenuItem', iptrw);
19125:     Breakpoint1', 'TMenuItem', iptrw);
19126:     SerialRS2321', 'TMenuItem', iptrw);
19127:     N14', 'TMenuItem', iptrw);
19128:     SynCSSyn1', 'TSynCSSyn', iptrw);
19129:     CSyntax2', 'TMenuItem', iptrw);
19130:     Calculator1', 'TMenuItem', iptrw);
19131:     tbtnSerial', 'TToolButton', iptrw);
19132:     ToolButton8', 'TToolButton', iptrw);
19133:     Tutorial151', 'TMenuItem', iptrw);
19134:     N15', 'TMenuItem', iptrw);
19135:     N16', 'TMenuItem', iptrw);
19136:     ControlBar1', 'TControlBar', iptrw);
19137:     ToolBar2', 'TToolBar', iptrw);
19138:     BtnOpen', 'TToolButton', iptrw);
19139:     BtnSave', 'TToolButton', iptrw);
19140:     BtnPrint', 'TToolButton', iptrw);
19141:     BtnColors', 'TToolButton', iptrw);
19142:     btnClassReport', 'TToolButton', iptrw);
19143:     BtnRotateRight', 'TToolButton', iptrw);
19144:     BtnFullSize', 'TToolButton', iptrw);
19145:     BtnFitToWindowSize', 'TToolButton', iptrw);
19146:     BtnZoomMinus', 'TToolButton', iptrw);
19147:     BtnZoomPlus', 'TToolButton', iptrw);
19148:     Panel1', 'TPanel', iptrw);
19149:     LabelBrettgroesse', 'TLabel', iptrw);
19150:     CB1SCList', 'TComboBox', iptrw);
19151:     ImageListNormal', 'TImageList', iptrw);
19152:     spbtnexplore', 'TSpeedButton', iptrw);
```

```
19153:     spbtnexample', 'TSpeedButton', iptrw);
19154:     spbsaveas', 'TSpeedButton', iptrw);
19155:     imglogobox', 'TImage', iptrw);
19156:     EnlargeFont1', 'TMenuItem', iptrw);
19157:     EnlargeFont2', 'TMenuItem', iptrw);
19158:     ShrinkFont1', 'TMenuItem', iptrw);
19159:     ThreadDemo1', 'TMenuItem', iptrw);
19160:     HEXEditor1', 'TMenuItem', iptrw);
19161:     HEXView1', 'TMenuItem', iptrw);
19162:     HEXInspect1', 'TMenuItem', iptrw);
19163:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
19164:     ExporttoHTML1', 'TMenuItem', iptrw);
19165:     ClassCount1', 'TMenuItem', iptrw);
19166:     HTMLOutput1', 'TMenuItem', iptrw);
19167:     HEXEditor2', 'TMenuItem', iptrw);
19168:     Minesweeper1', 'TMenuItem', iptrw);
19169:     N17', 'TMenuItem', iptrw);
19170:     PicturePuzzle1', 'TMenuItem', iptrw);
19171:     sbvclhelp', 'TSpeedButton', iptrw);
19172:     DependencyWalker1', 'TMenuItem', iptrw);
19173:     WebScanner1', 'TMenuItem', iptrw);
19174:     View1', 'TMenuItem', iptrw);
19175:     mnToolbar1', 'TMenuItem', iptrw);
19176:     mnStatusbar2', 'TMenuItem', iptrw);
19177:     mnConsole2', 'TMenuItem', iptrw);
19178:     mnCoolbar2', 'TMenuItem', iptrw);
19179:     mnSplitter2', 'TMenuItem', iptrw);
19180:     WebServer1', 'TMenuItem', iptrw);
19181:     Tutorial17Server1', 'TMenuItem', iptrw);
19182:     Tutorial18Arduino1', 'TMenuItem', iptrw);
19183:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
19184:     PerlSyntax1', 'TMenuItem', iptrw);
19185:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
19186:     PythonSyntax1', 'TMenuItem', iptrw);
19187:     DMathLibrary1', 'TMenuItem', iptrw);
19188:     IntfNavigator1', 'TMenuItem', iptrw);
19189:     EnlargeFontConsole1', 'TMenuItem', iptrw);
19190:     ShrinkFontConsole1', 'TMenuItem', iptrw);
19191:     SetInterfaceList1', 'TMenuItem', iptrw);
19192:     popintfList', 'TPopupMenu', iptrw);
19193:     intfAdd1', 'TMenuItem', iptrw);
19194:     intfDelete1', 'TMenuItem', iptrw);
19195:     intfRefactor1', 'TMenuItem', iptrw);
19196:     Defactor1', 'TMenuItem', iptrw);
19197:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
19198:     Tutorial20Regex', 'TMenuItem', iptrw);
19199:     N18', 'TMenuItem', iptrw);
19200:     ManualE1', 'TMenuItem', iptrw);
19201:     FullTextFinder1', 'TMenuItem', iptrw);
19202:     Move1', 'TMenuItem', iptrw);
19203:     FractalDemo1', 'TMenuItem', iptrw);
19204:     Tutorial21Android1', 'TMenuItem', iptrw);
19205:     Tutorial0Function1', 'TMenuItem', iptrw);
19206:     SimuLogBox1', 'TMenuItem', iptrw);
19207:     OpenExamples1', 'TMenuItem', iptrw);
19208:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
19209:     JavaScriptSyntax1', 'TMenuItem', iptrw);
19210:     Halt1', 'TMenuItem', iptrw);
19211:     CodeSearch1', 'TMenuItem', iptrw);
19212:     SynRubySyn1', 'TSynRubySyn', iptrw);
19213:     RubySyntax1', 'TMenuItem', iptrw);
19214:     Undo1', 'TMenuItem', iptrw);
19215:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
19216:     LinuxShellScript1', 'TMenuItem', iptrw);
19217:     Rename1', 'TMenuItem', iptrw);
19218:     spdcodesearch', 'TSpeedButton', iptrw);
19219:     Preview1', 'TMenuItem', iptrw);
19220:     Tutorial22Services1', 'TMenuItem', iptrw);
19221:     Tutorial23RealTime1', 'TMenuItem', iptrw);
19222:     Configuration1', 'TMenuItem', iptrw);
19223:     MP3Player1', 'TMenuItem', iptrw);
19224:     DLLSpy1', 'TMenuItem', iptrw);
19225:     SynURIOpener1', 'TSynURIOpener', iptrw);
19226:     SynURISyn1', 'TSynURISyn', iptrw);
19227:     URILinksClicks1', 'TMenuItem', iptrw);
19228:     EditReplace1', 'TMenuItem', iptrw);
19229:     GotoLine1', 'TMenuItem', iptrw);
19230:     ActiveLineColor1', 'TMenuItem', iptrw);
19231:     ConfigFile1', 'TMenuItem', iptrw);
19232:     Sort1Intflist', 'TMenuItem', iptrw);
19233:     Redo1', 'TMenuItem', iptrw);
19234:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
19235:     Tutorial25Configuration1', 'TMenuItem', iptrw);
19236:     IndentSelection1', 'TMenuItem', iptrw);
19237:     UnindentSection1', 'TMenuItem', iptrw);
19238:     SkyStyle1', 'TMenuItem', iptrw);
19239:     N19', 'TMenuItem', iptrw);
19240:     CountWords1', 'TMenuItem', iptrw);
19241:     imbookmarkimages', 'TImageList', iptrw);
```

```
19242:     Bookmark11', 'TMenuItem', iptrw);
19243:     N20', 'TMenuItem', iptrw);
19244:     Bookmark21', 'TMenuItem', iptrw);
19245:     Bookmark31', 'TMenuItem', iptrw);
19246:     Bookmark41', 'TMenuItem', iptrw);
19247:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
19248:
19249:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
19250:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
19251:     Procedure PSScriptCompile( Sender : TPSScript)
19252:     Procedure Compile1Click( Sender : TObject)
19253:     Procedure PSScriptExecute( Sender : TPSScript)
19254:     Procedure open1Click( Sender : TObject)
19255:     Procedure Save2Click( Sender : TObject)
19256:     Procedure Savebefore1Click( Sender : TObject)
19257:     Procedure Largefont1Click( Sender : TObject)
19258:     Procedure FormActivate( Sender : TObject)
19259:     Procedure SBytecode1Click( Sender : TObject)
19260:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
19261:     Procedure Saveas3Click( Sender : TObject)
19262:     Procedure Clear1Click( Sender : TObject)
19263:     Procedure Slinenumbers1Click( Sender : TObject)
19264:     Procedure About1Click( Sender : TObject)
19265:     Procedure Search1Click( Sender : TObject)
19266:     Procedure FormCreate( Sender : TObject)
19267:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
19268:                                                     var Action : TSynReplaceAction)
19269:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
19270:     Procedure WordWrap1Click( Sender : TObject)
19271:     Procedure SearchNext1Click( Sender : TObject)
19272:     Procedure Replace1Click( Sender : TObject)
19273:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
19274:     Procedure ShowInclude1Click( Sender : TObject)
19275:     Procedure Printout1Click( Sender : TObject)
19276:     Procedure mnuPrintFont1Click( Sender : TObject)
19277:     Procedure Include1Click( Sender : TObject)
19278:     Procedure FormDestroy( Sender : TObject)
19279:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
19280:     Procedure UpdateView1Click( Sender : TObject)
19281:     Procedure CodeCompletionList1Click( Sender : TObject)
19282:     Procedure SaveOutput1Click( Sender : TObject)
19283:     Procedure ExportClipboard1Click( Sender : TObject)
19284:     Procedure Close1Click( Sender : TObject)
19285:     Procedure Manual1Click( Sender : TObject)
19286:     Procedure LoadLastFile1Click( Sender : TObject)
19287:     Procedure Memo1Change( Sender : TObject)
19288:     Procedure Decompile1Click( Sender : TObject)
19289:     Procedure StepInto1Click( Sender : TObject)
19290:     Procedure StepOut1Click( Sender : TObject)
19291:     Procedure Reset1Click( Sender : TObject)
19292:     Procedure cedebugAfterExecute( Sender : TPSScript)
19293:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
19294:     Procedure cedebugCompile( Sender : TPSScript)
19295:     Procedure cedebugExecute( Sender : TPSScript)
19296:     Procedure cedebugIdle( Sender : TObject)
19297:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
19298:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
19299:     Procedure BreakPointMenuClick( Sender : TObject)
19300:     Procedure DebugRun1Click( Sender : TObject)
19301:     Procedure tutorial4Click( Sender : TObject)
19302:     Procedure ImportfromClipboard1Click( Sender : TObject)
19303:     Procedure ImportfromClipboard2Click( Sender : TObject)
19304:     Procedure tutorial1Click( Sender : TObject)
19305:     Procedure ShowSpecChars1Click( Sender : TObject)
19306:     Procedure StatusBar1DblClick( Sender : TObject)
19307:     Procedure PSScriptLine( Sender : TObject)
19308:     Procedure OpenDirectory1Click( Sender : TObject)
19309:     Procedure procMessClick( Sender : TObject)
19310:     Procedure tbtnUseCaseClick( Sender : TObject)
19311:     Procedure EditFont1Click( Sender : TObject)
19312:     Procedure tutorial21Click( Sender : TObject)
19313:     Procedure tutorial31Click( Sender : TObject)
19314:     Procedure HTMLSyntax1Click( Sender : TObject)
19315:     Procedure ShowInterfaces1Click( Sender : TObject)
19316:     Procedure Tutorial5Click( Sender : TObject)
19317:     Procedure ShowLastException1Click( Sender : TObject)
19318:     Procedure PlayMP31Click( Sender : TObject)
19319:     Procedure AllFunctionsList1Click( Sender : TObject)
19320:     Procedure texSyntax1Click( Sender : TObject)
19321:     Procedure GetEMails1Click( Sender : TObject)
19322:     procedure DelphiSite1Click(Sender: TObject);
19323:     procedure TerminalStyle1Click(Sender: TObject);
19324:     procedure ReadOnly1Click(Sender: TObject);
19325:     procedure ShellStyle1Click(Sender: TObject);
19326:     procedure Console1Click(Sender: TObject);     //3.2
19327:     procedure BigScreen1Click(Sender: TObject);
19328:     procedure Tutorial91Click(Sender: TObject);
19329:     procedure SaveScreenshotClick(Sender: TObject);
19330:     procedure Tutorial101Click(Sender: TObject);
```

```
19331:     procedure SQLSyntax1Click(Sender: TObject);
19332:     procedure XMLSyntax1Click(Sender: TObject);
19333:     procedure ComponentCount1Click(Sender: TObject);
19334:     procedure NewInstance1Click(Sender: TObject);
19335:     procedure CSyntax1Click(Sender: TObject);
19336:     procedure Tutorial6Click(Sender: TObject);
19337:     procedure New1Click(Sender: TObject);
19338:     procedure AllObjectsList1Click(Sender: TObject);
19339:     procedure LoadBytecode1Click(Sender: TObject);
19340:     procedure CipherFile1Click(Sender: TObject);  //V3.5
19341:     procedure NewInstance1Click(Sender: TObject);
19342:     procedure toolbtnTutorialClick(Sender: TObject);
19343:     procedure Memory1Click(Sender: TObject);
19344:     procedure JavaSyntax1Click(Sender: TObject);
19345:     procedure SyntaxCheck1Click(Sender: TObject);
19346:     procedure ScriptExplorer1Click(Sender: TObject);
19347:     procedure FormOutput1Click(Sender: TObject);  //V3.6
19348:     procedure GotoEnd1Click(Sender: TObject);
19349:     procedure AllResourceList1Click(Sender: TObject);
19350:     procedure tbtn6resClick(Sender: TObject);   //V3.7
19351:     procedure Info1Click(Sender: TObject);
19352:     procedure Tutorial10Statistics1Click(Sender: TObject);
19353:     procedure Tutorial11Forms1Click(Sender: TObject);
19354:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
19355:     procedure ResourceExplore1Click(Sender: TObject);
19356:     procedure Info1Click(Sender: TObject);
19357:     procedure CryptoBox1Click(Sender: TObject);
19358:     procedure ModulesCount1Click(Sender: TObject);
19359:     procedure N4GewinntGame1Click(Sender: TObject);
19360:     procedure PHPSyntax1Click(Sender: TObject);
19361:     procedure SerialRS2321Click(Sender: TObject);
19362:     procedure CSyntax2Click(Sender: TObject);
19363:     procedure Calculator1Click(Sender: TObject);
19364:     procedure Tutorial13Ciphering1Click(Sender: TObject);
19365:     procedure Tutorial14Async1Click(Sender: TObject);
19366:     procedure PHPSyntax1Click(Sender: TObject);
19367:     procedure BtnZoomPlusClick(Sender: TObject);
19368:     procedure BtnZoomMinusClick(Sender: TObject);
19369:     procedure btnClassReportClick(Sender: TObject);
19370:     procedure ThreadDemo1Click(Sender: TObject);
19371:     procedure HEXView1Click(Sender: TObject);
19372:     procedure ExporttoHTML1Click(Sender: TObject);
19373:     procedure Minesweeper1Click(Sender: TObject);
19374:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
19375:     procedure sbvclhelpClick(Sender: TObject);
19376:     procedure DependencyWalker1Click(Sender: TObject);
19377:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
19378:     procedure WebScanner1Click(Sender: TObject);
19379:     procedure mnToolbar1Click(Sender: TObject);
19380:     procedure mnStatusbar2Click(Sender: TObject);
19381:     procedure mnConsole2Click(Sender: TObject);
19382:     procedure mnCoolbar2Click(Sender: TObject);
19383:     procedure mnSplitter2Click(Sender: TObject);
19384:     procedure WebServer1Click(Sender: TObject);
19385:     procedure PerlSyntax1Click(Sender: TObject);
19386:     procedure PythonSyntax1Click(Sender: TObject);
19387:     procedure DMathLibrary1Click(Sender: TObject);
19388:     procedure IntfNavigator1Click(Sender: TObject);
19389:     procedure FullTextFinder1Click(Sender: TObject);
19390:     function AppName: string;
19391:     function ScriptName: string;
19392:     function LastName: string;
19393:     procedure FractalDemo1Click(Sender: TObject);
19394:     procedure SimuLogBox1Click(Sender: TObject);
19395:     procedure OpenExamples1Click(Sender: TObject);
19396:     procedure Halt1Click(Sender: TObject);
19397:     procedure Stop;
19398:     procedure CodeSearch1Click(Sender: TObject);
19399:     procedure RubySyntax1Click(Sender: TObject);
19400:     procedure Undo1Click(Sender: TObject);
19401:     procedure LinuxShellScript1Click(Sender: TObject);
19402:     procedure WebScannerDirect(urls: string);
19403:     procedure WebScanner(urls: string);
19404:     procedure LoadInterfaceList2;
19405:     procedure DLLSpy1Click(Sender: TObject);
19406:     procedure Memo1DblClick(Sender: TObject);
19407:     procedure URILinksClicks1Click(Sender: TObject);
19408:     procedure GotoLine1Click(Sender: TObject);
19409:     procedure ConfigFile1Click(Sender: TObject);
19410:     Procedure Sort1IntflistClick( Sender : TObject)
19411:     Procedure Redo1Click( Sender : TObject)
19412:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
19413:     Procedure IndentSelection1Click( Sender : TObject)
19414:     Procedure UnindentSection1Click( Sender : TObject)
19415:     Procedure SkyStyle1Click( Sender : TObject)
19416:     Procedure CountWords1Click( Sender : TObject)
19417:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
19418:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
19419:     Procedure Bookmark11Click( Sender : TObject)
```

```
19420:     Procedure Bookmark21Click( Sender : TObject)
19421:     Procedure Bookmark31Click( Sender : TObject)
19422:     Procedure Bookmark41Click( Sender : TObject)
19423:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
19424:     'STATMemoryReport', 'boolean', iptrw);
19425:     'IPPort', 'integer', iptrw);
19426:     'COMPort', 'integer', iptrw);
19427:     'lbintflist', 'TListBox', iptrw);
19428:     Function GetStatChange : boolean
19429:     Procedure SetStatChange( vstat : boolean)
19430:     Function GetActFileName : string
19431:     Procedure SetActFileName( vname : string)
19432:     Function GetLastFileName : string
19433:     Procedure SetLastFileName( vname : string)
19434:     Procedure WebScannerDirect( urls : string)
19435:     Procedure LoadInterfaceList2
19436:     Function GetStatExecuteShell : boolean
19437:     Procedure DoEditorExecuteCommand( EditorCommand : word)
19438:     function GetActiveLineColor: TColor
19439:     procedure SetActiveLineColor(acolor: TColor)
19440:     procedure ScriptListbox1Click(Sender: TObject);
19441:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
19442:     procedure EnlargeGutter1Click(Sender: TObject);
19443:     procedure Tetris1Click(Sender: TObject);
19444:     procedure ToDoList1Click(Sender: TObject);
19445:     procedure ProcessList1Click(Sender: TObject);
19446:     procedure MetricReport1Click(Sender: TObject);
19447:     procedure ProcessList1Click(Sender: TObject);
19448:     procedure TCPSockets1Click(Sender: TObject);
19449:     procedure ConfigUpdate1Click(Sender: TObject);
19450:     procedure ADOWorkbench1Click(Sender: TObject);
19451:     procedure SocketServer1Click(Sender: TObject);
19452:     procedure FormDemo1Click(Sender: TObject);
19453:     procedure Richedit1Click(Sender: TObject);
19454:     procedure SimpleBrowser1Click(Sender: TObject);
19455:     procedure DOSShell1Click(Sender: TObject);
19456:     procedure SynExport1Click(Sender: TObject);
19457:     procedure ExporttoRTF1Click(Sender: TObject);
19458:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
19459:     procedure SOAPTester1Click(Sender: TObject);
19460:     procedure Sniffer1Click(Sender: TObject);
19461:     procedure AutoDetectSyntax1Click(Sender: TObject);
19462:     procedure FPlot1Click(Sender: TObject);
19463:     procedure PasStyle1Click(Sender: TObject);
19464:     procedure Tutorial183RGBLED1Click(Sender: TObject);
19465:     procedure Reversi1Click(Sender: TObject);
19466:     procedure ManualmaXbox1Click(Sender: TObject);
19467:     procedure BlaisePascalMagazine1Click(Sender: TObject);
19468:     procedure AddToDo1Click(Sender: TObject);
19469:     procedure CreateGUID1Click(Sender: TObject);
19470:     procedure Tutorial27XML1Click(Sender: TObject);
19471:     procedure CreateDLLStub1Click(Sender: TObject);
19472:     procedure Tutorial28DLL1Click(Sender: TObject);');
19473:     procedure ResetKeyPressed;');
19474:     procedure FileChanges1Click(Sender: TObject);');
19475:     procedure OpenGLTry1Click(Sender: TObject);');
19476:     procedure AllUnitList1Click(Sender: TObject);');
19477:     procedure Tutorial29UMLClick(Sender: TObject);
19478:     procedure CreateHeader1Click(Sender: TObject);
19479:
19480: //-----------------------------------------------------------------------------
19481: //**************mX4 Editor SynEdit Tools API ********************************
19482: //-----------------------------------------------------------------------------
19483: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
19484: begin
19485:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
19486:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
19487:     Constructor Create( AOwner : TComponent)
19488:     SelStart', 'Integer', iptrw);
19489:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
19490:     Procedure UpdateCaret
19491:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
19492:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
19493:     Procedure BeginUndoBlock
19494:     Procedure BeginUpdate
19495:     Function CaretInView : Boolean
19496:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
19497:     Procedure Clear
19498:     Procedure ClearAll
19499:     Procedure ClearBookMark( BookMark : Integer)
19500:     Procedure ClearSelection
19501:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
19502:     Procedure ClearUndo
19503:     Procedure CopyToClipboard
19504:     Procedure CutToClipboard
19505:     Procedure DoCopyToClipboard( const SText : string)
19506:     Procedure EndUndoBlock
19507:     Procedure EndUpdate
19508:     Procedure EnsureCursorPosVisible
```

```
19509:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
19510:     Procedure FindMatchingBracket
19511:     Function GetMatchingBracket : TBufferCoord
19512:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
19513:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
19514:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
19515:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
19516:               : TSynHighlighterAttributes) : boolean
19517:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
19518:               var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
19519:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
19520:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
19521:     Procedure GotoBookMark( BookMark : Integer)
19522:     Procedure GotoLineAndCenter( ALine : Integer)
19523:     Function IdentChars : TSynIdentChars
19524:     Procedure InvalidateGutter
19525:     Procedure InvalidateGutterLine( aLine : integer)
19526:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
19527:     Procedure InvalidateLine( Line : integer)
19528:     Procedure InvalidateLines( FirstLine, LastLine : integer)
19529:     Procedure InvalidateSelection
19530:     Function IsBookmark( BookMark : integer) : boolean
19531:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
19532:     Procedure LockUndo
19533:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
19534:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
19535:     Function LineToRow( aLine : integer) : integer
19536:     Function RowToLine( aRow : integer) : integer
19537:     Function NextWordPos : TBufferCoord
19538:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
19539:     Procedure PasteFromClipboard
19540:     Function WordStart : TBufferCoord
19541:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
19542:     Function WordEnd : TBufferCoord
19543:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
19544:     Function PrevWordPos : TBufferCoord
19545:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
19546:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
19547:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
19548:     Procedure Redo
19549:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
19550:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
19551:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
19552:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
19553:     Procedure SelectAll
19554:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
19555:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
19556:     Procedure SetDefaultKeystrokes
19557:     Procedure SetSelWord
19558:     Procedure SetWordBlock( Value : TBufferCoord)
19559:     Procedure Undo
19560:     Procedure UnlockUndo
19561:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
19562:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
19563:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
19564:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
19565:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
19566:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
19567:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
19568:     Procedure AddFocusControl( aControl : TWinControl)
19569:     Procedure RemoveFocusControl( aControl : TWinControl)
19570:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
19571:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
19572:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
19573:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
19574:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
19575:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
19576:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
19577:     Procedure RemoveLinesPointer
19578:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
19579:     Procedure UnHookTextBuffer
19580:     BlockBegin', 'TBufferCoord', iptrw);
19581:     BlockEnd', 'TBufferCoord', iptrw);
19582:     CanPaste', 'Boolean', iptr);
19583:     CanRedo', 'boolean', iptr);
19584:     CanUndo', 'boolean', iptr);
19585:     CaretX', 'Integer', iptrw);
19586:     CaretY', 'Integer', iptrw);
19587:     CaretXY', 'TBufferCoord', iptrw);
19588:     ActiveLineColor', 'TColor', iptrw);
19589:     DisplayX', 'Integer', iptr);
19590:     DisplayY', 'Integer', iptr);
19591:     DisplayXY', 'TDisplayCoord', iptr);
19592:     DisplayLineCount', 'integer', iptr);
19593:     CharsInWindow', 'Integer', iptr);
19594:     CharWidth', 'integer', iptr);
19595:     Font', 'TFont', iptrw);
19596:     GutterWidth', 'Integer', iptr);
19597:     Highlighter', 'TSynCustomHighlighter', iptrw);
```

```
19598:     LeftChar', 'Integer', iptrw);
19599:     LineHeight', 'integer', iptr);
19600:     LinesInWindow', 'Integer', iptr);
19601:     LineText', 'string', iptrw);
19602:     Lines', 'TStrings', iptrw);
19603:     Marks', 'TSynEditMarkList', iptr);
19604:     MaxScrollWidth', 'integer', iptrw);
19605:     Modified', 'Boolean', iptrw);
19606:     PaintLock', 'Integer', iptr);
19607:     ReadOnly', 'Boolean', iptrw);
19608:     SearchEngine', 'TSynEditSearchCustom', iptrw);
19609:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
19610:     SelTabBlock', 'Boolean', iptr);
19611:     SelTabLine', 'Boolean', iptr);
19612:     SelText', 'string', iptrw);
19613:     StateFlags', 'TSynStateFlags', iptr);
19614:     Text', 'string', iptrw);
19615:     TopLine', 'Integer', iptrw);
19616:     WordAtCursor', 'string', iptr);
19617:     WordAtMouse', 'string', iptr);
19618:     UndoList', 'TSynEditUndoList', iptr);
19619:     RedoList', 'TSynEditUndoList', iptr);
19620:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
19621:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
19622:     BorderStyle', 'TSynBorderStyle', iptrw);
19623:     ExtraLineSpacing', 'integer', iptrw);
19624:     Gutter', 'TSynGutter', iptrw);
19625:     HideSelection', 'boolean', iptrw);
19626:     InsertCaret', 'TSynEditCaretType', iptrw);
19627:     InsertMode', 'boolean', iptrw);
19628:     IsScrolling', 'Boolean', iptr);
19629:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
19630:     MaxUndo', 'Integer', iptrw);
19631:     Options', 'TSynEditorOptions', iptrw);
19632:     OverwriteCaret', 'TSynEditCaretType', iptrw);
19633:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
19634:     ScrollHintColor', 'TColor', iptrw);
19635:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
19636:     ScrollBars', 'TScrollStyle', iptrw);
19637:     SelectedColor', 'TSynSelectedColor', iptrw);
19638:     SelectionMode', 'TSynSelectionMode', iptrw);
19639:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
19640:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
19641:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
19642:     WordWrapGlyph', 'TSynGlyph', iptrw);
19643:     OnChange', 'TNotifyEvent', iptrw);
19644:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
19645:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
19646:     OnContextHelp', 'TContextHelpEvent', iptrw);
19647:     OnDropFiles', 'TDropFilesEvent', iptrw);
19648:     OnGutterClick', 'TGutterClickEvent', iptrw);
19649:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
19650:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
19651:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
19652:     OnPaint', 'TPaintEvent', iptrw);
19653:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
19654:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
19655:     OnReplaceText', 'TReplaceTextEvent', iptrw);
19656:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
19657:     OnStatusChange', 'TStatusChangeEvent', iptrw);
19658:     OnPaintTransient', 'TPaintTransient', iptrw);
19659:     OnScroll', 'TScrollEvent', iptrw);
19660:   end;
19661: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
19662: Function GetPlaceableHighlighters : TSynHighlighterList
19663: Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
19664: Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
19665: Procedure GetEditorCommandValues( Proc : TGetStrProc)
19666: Procedure GetEditorCommandExtended( Proc : TGetStrProc)
19667: Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
19668: Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
19669: Function ConvertCodeStringToExtended( AString : String) : String
19670: Function ConvertExtendedToCodeString( AString : String) : String
19671: Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
19672: Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
19673: Function IndexToEditorCommand( const AIndex : Integer) : Integer
19674:
19675:  TSynEditorOption = (
19676:    eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
19677:    eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
19678:                               // preceding line
19679:    eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
19680:    eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
19681:                               //direction any more
19682:    eoDragDropEditing,         //Allows to select a block of text and drag it within document to another
19683:                               // location
19684:    eoDropFiles,               //Allows the editor accept OLE file drops
19685:    eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
19686:    eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
```

```
19687:     eoGroupUndo,               //When undoing/redoing actions, handle all continous changes the same kind
19688:                                // in one call
19689:                                //instead undoing/redoing each command separately
19690:     eoHalfPageScroll,          //When scrolling with page-up and page-down commands, only scroll a half
19691:                                //page at a time
19692:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
19693:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
19694:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
19695:     eoNoCaret,                 //Makes it so the caret is never visible
19696:     eoNoSelection,             //Disables selecting text
19697:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
19698:     eoScrollByOneLess,         //Forces scrolling to be one less
19699:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
19700:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
19701:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
19702:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
19703:     eoShowSpecialChars,        //Shows the special Characters
19704:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
19705:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
19706:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
19707:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
19708:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
19709:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
19710:
19711:     ********************************Important Editor Short Cuts*****************************);
19712:     Double click to select a word and count words with highlightning.
19713:     Triple click to select a line.
19714:     CTRL+SHIFT+click to extend a selection.
19715:     Drag with the ALT key down to select columns of text !!!
19716:     Drag and drop is supported.
19717:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
19718:     Type CTRL+A to select all.
19719:     Type CTRL+N to set a new line.
19720:     Type CTRL+T to delete a line or token. //Tokenizer
19721:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
19722:     Type CTRL+Shift+T to add ToDo in line and list.
19723:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
19724:     Type CTRL[0..9] to jump or get to bookmarks.
19725:     Type Home to position cursor at beginning of current line and End to position it at end of line.
19726:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
19727:     Page Up and Page Down work as expected.
19728:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
19729:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
19730:
19731: {$ Short Key Positions Ctrl<A-Z>: }
19732: def
19733:    <A> Select All
19734:    <B> Count Words
19735:    <C> Copy
19736:    <D> Internet Start
19737:    <E> Script List
19738:    <F> Find
19739:    <G> Goto
19740:    <H> Mark Line
19741:    <I> Interface List
19742:    <J> Code Completion
19743:    <K> Console
19744:    <L> Interface List Box
19745:    <M> Font Larger -
19746:    <N> New Line
19747:    <O> Open File
19748:    <P> Font Smaller +
19749:    <Q> Quit
19750:    <R> Replace
19751:    <S> Save!
19752:    <T> Delete Line
19753:    <U> Use Case Editor
19754:    <V> Paste
19755:    <W> URI Links
19756:    <X> Reserved for coding use internal
19757:    <Y> Delete Line
19758:    <Z> Undo
19759:
19760: ref
19761:    F1 Help
19762:    F2 Syntax Check
19763:    F3 Search Next
19764:    F4 New Instance
19765:    F5 Line Mark /Breakpoint
19766:    F6 Goto End
19767:    F7 Debug Step Into
19768:    F8 Debug Step Out
19769:    F9 Compile
19770:    F10 Menu
19771:    F11 Word Count Highlight
19772:    F12 Reserved for coding use internal
19773:
19774:  def ReservedWords: array[0..82] of string =
19775:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
```

```
19776:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
19777:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
19778:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
19779:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
19780:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
19781:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
19782:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
19783:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
19784:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
19785:      'public', 'published','def','ref','using','typedef','memo1','memo2','doc','maxform1';
19786:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
19787:
19788: //-------------------------------------------------------------------------
19789: //*************End of mX4 Public  Tools API *********************************
19790: //-------------------------------------------------------------------------
19791:
19792: Amount of Functions: 12527
19793: Amount of Procedures: 7767
19794: Amount of Constructors: 1264
19795: Totals of Calls: 21558
19796: SHA1: Win 3.9.9.95 A8BE7AD5B70ECAF9797C5C9E42A3CD40E903145F
19797:
19798: ********************************************************
19799:  Doc Short Manual with 50 Tips!
19800: ********************************************************
19801: - Install: just save your maxboxdef.ini before and then extract the zip file!
19802: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
19803: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
19804: - Menu: With <Crtl><F3> you can search for code on examples
19805: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
19806: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
19807: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
19808:
19809: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
19810: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
19811: - Context Menu: You can printout your scripts as a pdf-file or html-export
19812: - Context: You do have a context menu with the right mouse click
19813:
19814: - Menu: With the UseCase Editor you can convert graphic formats too.
19815: - Menu: On menu Options you find Addons as compiled scripts
19816: - IDE: You don't need a mouse to handle maXbox, use shortcuts
19817: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
19818: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
19819: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
19820:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
19821:
19822: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
19823: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
19824: - Code: If you code a loop till key-pressed use function: isKeyPressed;
19825: - Code: Macro set the macros #name,, #paAdministratorth, #file,startmaxbox_extract_funclist399.txter25.pdf
19826: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
19827: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
19828:          to delete and Click and mark to drag a bookmark
19829: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
19830: - IDE: A file info with system and script information you find in menu Program/Information
19831: - IDE: After change the config file in help you can update changes in menu Help/Config Update
19832: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
19833: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
19834: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
19835: - Editor: Set Bookmarks to check your work in app or code
19836: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
19837: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
19838: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
19839:
19840: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
19841: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
19842: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
19843: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
19844: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
19845: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
19846: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
19847: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
19848:
19849: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
19850: - Add on when no browser is available start /Options/Add ons/Easy Browser
19851: - Add on SOAP Tester with SOP POST File
19852: - Add on IP Protocol Sniffer with List View
19853: - Add on OpenGL mX Robot Demo for android
19854:
19855: - Menu: Help/Tools as a Tool Section with DOS Opener
19856: - Menu Editor: export the code as RTF File
19857: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
19858: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
19859: - Context: Auto Detect of Syntax depending on file extension
19860: - Code: some Windows API function start with w in the name like wGetAtomName();
19861: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
19862: - IDE File Check with menu ..View/File Changes/...
19863: - Context: Create a Header with Create Header in Navigator List at right window
19864: - Code: use  SysErrorMessage to get a real Error Description, Ex.
```

```
19865:         RemoveDir('c:\NoSuchFolder') ;
19866:         writeln('System Error Message: '+ SysErrorMessage(GetLastError));
19867:
19868:
19869: - using DLL example in maXbox: //function: {*****************************************}
19870:    Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
19871:                                  cb: DWORD): BOOL; //stdcall;;
19872:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
19873:    Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
19874:      External  'OpenProcess@kernel32.dll stdcall';
19875:
19876:
19877: PCT Precompile Technology , mX4 ScriptStudio
19878: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
19879: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
19880: emax layers: system-package-component-unit-class-function-block
19881: new keywords def ref using maXCalcF
19882: UML: use case act class state seq pac comp dep - lib lab
19883: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
19884: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
19885: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
19886: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
19887: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
19888: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
19889: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
19890: https://unibe-ch.academia.edu/MaxKleiner
19891: www.slideshare.net/maxkleiner1
19892: http://www.scribd.com/max_kleiner
19893: http://www.delphiforfun.org/Programs/Utilities/index.htm
19894:
19895:
19896:
19897:
19898: ************************************************************
19899:  unit List asm internal end
19900: ************************************************************
19901: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
19902: 02 unit SIRegister_IdStrings                   //Indy Sockets
19903: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
19904: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
19905: 05 unit IFSI_WinForm1puzzle;                   //maXbox
19906: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
19907: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
19908: 08 unit RIRegister_MathMax_Routines(exec);    //Jedi & Delphi
19909: 09 unit RIRegister_IdGlobal_Routines(exec);   //Indy Sockets
19910: 10 unit RIRegister_SysUtils_Routines(Exec);   //Delphi
19911: 11 unit uPSI_IdTCPConnection;                 //Indy some functions
19912: 12 unit uPSCompiler.pas;                      //PS kernel functions
19913: 13 unit uPSI_DBCommon;                       //DB Common_Routines and Types
19914: 14 unit uPSI_Printers.pas                     //Delphi VCL
19915: 15 unit uPSI_MPlayer.pas                     //Delphi VCL
19916: 16 unit uPSC_comobj;                         //COM Functions
19917: 17 unit uPSI_Clipbrd;                        //Delphi VCL
19918: 18 unit Filectrl in IFSI_SysUtils_max;      //VCL Runtime
19919: 19 unit uPSI_SqlExpr;                       //DBX3
19920: 20 unit uPSI_ADODB;                         //ADODB
19921: 21 unit uPSI_StrHlpr;                       //String Helper Routines
19922: 22 unit uPSI_DateUtils;                     //Expansion to DateTimeLib
19923: 23 unit uPSI_FileUtils;                     //Expansion to Sys/File Utils
19924: 24 unit JUtils / gsUtils;                   //Jedi / Metabase
19925: 25 unit JvFunctions_max;                    //Jedi Functions
19926: 26 unit HTTPParser;                         //Delphi VCL
19927: 27 unit HTTPUtil;                          //Delphi VCL
19928: 28 unit uPSI_XMLUtil;                      //Delphi VCL
19929: 29 unit uPSI_SOAPHTTPClient;               //Delphi VCL SOAP WebService V3.5
19930: 30 unit uPSI_Contnrs;                      //Delphi RTL Container of Classes
19931: 31 unit uPSI_MaskUtils;                    //RTL Edit and Mask functions
19932: 32 unit uPSI_MyBigInt;                     //big integer class with Math
19933: 33 unit uPSI_ConvUtils;                    //Delphi VCL Conversions engine
19934: 34 unit Types_Variants;                    //Delphi\Win32\rtl\sys
19935: 35 unit uPSI_IdHashSHA1;                   //Indy Crypto Lib
19936: 36 unit uPSI_IdHashMessageDigest          //Indy Crypto;
19937: 37 unit uPSI_IdASN1Util;                   //Indy ASN1Utility Routines;
19938: 38 unit uPSI_IdLogFile;                    //Indy Logger from LogBase
19939: 39 unit uPSI_IdIcmpClient;                 //Indy Ping ICMP
19940: 40 unit uPSI_IdHashMessageDigest_max      //Indy Crypto &OpenSSL;
19941: 41 unit uPSI_FileCtrl;                     //Delphi RTL
19942: 42 unit uPSI_Outline;                      //Delphi VCL
19943: 43 unit uPSI_ScktComp;                     //Delphi RTL
19944: 44 unit uPSI_Calendar;                     //Delphi VCL
19945: 45 unit uPSI_VListView                     //VListView;
19946: 46 unit uPSI_DBGrids;                      //Delphi VCL
19947: 47 unit uPSI_DBCtrls;                      //Delphi VCL
19948: 48 unit ide_debugoutput;                   //maXbox
19949: 49 unit uPSI_ComCtrls;                     //Delphi VCL
19950: 50 unit uPSC_stdctrls+;                    //Delphi VCL
19951: 51 unit uPSI_Dialogs;                      //Delphi VCL
19952: 52 unit uPSI_StdConvs;                     //Delphi RTL
19953: 53 unit uPSI_DBClient;                     //Delphi RTL
```

```
19954: 54 unit uPSI_DBPlatform;                    //Delphi RTL
19955: 55 unit uPSI_Provider;                      //Delphi RTL
19956: 56 unit uPSI_FMTBcd;                        //Delphi RTL
19957: 57 unit uPSI_DBCGrids;                      //Delphi VCL
19958: 58 unit uPSI_CDSUtil;                       //MIDAS
19959: 59 unit uPSI_VarHlpr;                       //Delphi RTL
19960: 60 unit uPSI_ExtDlgs;                       //Delphi VCL
19961: 61 unit sdpStopwatch;                       //maXbox
19962: 62 unit uPSI_JclStatistics;                 //JCL
19963: 63 unit uPSI_JclLogic;                      //JCL
19964: 64 unit uPSI_JclMiscel;                     //JCL
19965: 65 unit uPSI_JclMath_max;                   //JCL RTL
19966: 66 unit uPSI_uTPLb_StreamUtils;             //LockBox 3
19967: 67 unit uPSI_MathUtils;                     //BCB
19968: 68 unit uPSI_JclMultimedia;                 //JCL
19969: 69 unit uPSI_WideStrUtils;                  //Delphi API/RTL
19970: 70 unit uPSI_GraphUtil;                     //Delphi RTL
19971: 71 unit uPSI_TypeTrans;                     //Delphi RTL
19972: 72 unit uPSI_HTTPApp;                       //Delphi VCL
19973: 73 unit uPSI_DBWeb;                         //Delphi VCL
19974: 74 unit uPSI_DBBdeWeb;                      //Delphi VCL
19975: 75 unit uPSI_DBXpressWeb;                   //Delphi VCL
19976: 76 unit uPSI_ShadowWnd;                     //Delphi VCL
19977: 77 unit uPSI_ToolWin;                       //Delphi VCL
19978: 78 unit uPSI_Tabs;                          //Delphi VCL
19979: 79 unit uPSI_JclGraphUtils;                 //JCL
19980: 80 unit uPSI_JclCounter;                    //JCL
19981: 81 unit uPSI_JclSysInfo;                    //JCL
19982: 82 unit uPSI_JclSecurity;                   //JCL
19983: 83 unit uPSI_JclFileUtils;                  //JCL
19984: 84 unit uPSI_IdUserAccounts;                //Indy
19985: 85 unit uPSI_IdAuthentication;              //Indy
19986: 86 unit uPSI_uTPLb_AES;                     //LockBox 3
19987: 87 unit uPSI_IdHashSHA1;                    //LockBox 3
19988: 88 unit uTPLb_BlockCipher;                  //LockBox 3
19989: 89 unit uPSI_ValEdit.pas;                   //Delphi VCL
19990: 90 unit uPSI_JvVCLUtils;                    //JCL
19991: 91 unit uPSI_JvDBUtil;                      //JCL
19992: 92 unit uPSI_JvDBUtils;                     //JCL
19993: 93 unit uPSI_JvAppUtils;                    //JCL
19994: 94 unit uPSI_JvCtrlUtils;                   //JCL
19995: 95 unit uPSI_JvFormToHtml;                  //JCL
19996: 96 unit uPSI_JvParsing;                     //JCL
19997: 97 unit uPSI_SerDlgs;                       //Toolbox
19998: 98 unit uPSI_Serial;                        //Toolbox
19999: 99 unit uPSI_JvComponent;                   //JCL
20000: 100 unit uPSI_JvCalc;                      //JCL
20001: 101 unit uPSI_JvBdeUtils;                  //JCL
20002: 102 unit uPSI_JvDateUtil;                  //JCL
20003: 103 unit uPSI_JvGenetic;                   //JCL
20004: 104 unit uPSI_JclBase;                     //JCL
20005: 105 unit uPSI_JvUtils;                     //JCL
20006: 106 unit uPSI_JvStrUtil;                   //JCL
20007: 107 unit uPSI_JvStrUtils;                  //JCL
20008: 108 unit uPSI_JvFileUtil;                  //JCL
20009: 109 unit uPSI_JvMemoryInfos;               //JCL
20010: 110 unit uPSI_JvComputerInfo;              //JCL
20011: 111 unit uPSI_JvgCommClasses;              //JCL
20012: 112 unit uPSI_JvgLogics;                   //JCL
20013: 113 unit uPSI_JvLED;                       //JCL
20014: 114 unit uPSI_JvTurtle;                    //JCL
20015: 115 unit uPSI_SortThds; unit uPSI_ThSort;  //maXbox
20016: 116 unit uPSI_JvgUtils;                    //JCL
20017: 117 unit uPSI_JvExprParser;                //JCL
20018: 118 unit uPSI_HexDump;                     //Borland
20019: 119 unit uPSI_DBLogDlg;                    //VCL
20020: 120 unit uPSI_SqlTimSt;                    //RTL
20021: 121 unit uPSI_JvHtmlParser;                //JCL
20022: 122 unit uPSI_JvgXMLSerializer;           //JCL
20023: 123 unit uPSI_JvJCLUtils;                  //JCL
20024: 124 unit uPSI_JvStrings;                   //JCL
20025: 125 unit uPSI_uTPLb_IntegerUtils;          //TurboPower
20026: 126 unit uPSI_uTPLb_HugeCardinal;          //TurboPower
20027: 127 unit uPSI_uTPLb_HugeCardinalUtils;     //TurboPower
20028: 128 unit uPSI_SynRegExpr;                  //SynEdit
20029: 129 unit uPSI_StUtils;                     //SysTools4
20030: 130 unit uPSI_StToHTML;                    //SysTools4
20031: 131 unit uPSI_StStrms;                     //SysTools4
20032: 132 unit uPSI_StFIN;                       //SysTools4
20033: 133 unit uPSI_StAstroP;                    //SysTools4
20034: 134 unit uPSI_StStat;                      //SysTools4
20035: 135 unit uPSI_StNetCon;                    //SysTools4
20036: 136 unit uPSI_StDecMth;                    //SysTools4
20037: 137 unit uPSI_StOStr;                      //SysTools4
20038: 138 unit uPSI_StPtrns;                     //SysTools4
20039: 139 unit uPSI_StNetMsg;                    //SysTools4
20040: 140 unit uPSI_StMath;                      //SysTools4
20041: 141 unit uPSI_StExpEng;                    //SysTools4
20042: 142 unit uPSI_StCRC;                       //SysTools4
```

```
20043: 143 unit uPSI_StExport,                    //SysTools4
20044: 144 unit uPSI_StExpLog,                    //SysTools4
20045: 145 unit uPSI_ActnList;                    //Delphi VCL
20046: 146 unit uPSI_jpeg;                        //Borland
20047: 147 unit uPSI_StRandom;                    //SysTools4
20048: 148 unit uPSI_StDict;                      //SysTools4
20049: 149 unit uPSI_StBCD;                       //SysTools4
20050: 150 unit uPSI_StTxtDat;                    //SysTools4
20051: 151 unit uPSI_StRegEx;                     //SysTools4
20052: 152 unit uPSI_IMouse;                      //VCL
20053: 153 unit uPSI_SyncObjs;                    //VCL
20054: 154 unit uPSI_AsyncCalls;                  //Hausladen
20055: 155 unit uPSI_ParallelJobs;                //Saraiva
20056: 156 unit uPSI_Variants;                    //VCL
20057: 157 unit uPSI_VarCmplx;                    //VCL Wolfram
20058: 158 unit uPSI_DTDSchema;                   //VCL
20059: 159 unit uPSI_ShLwApi;                     //Brakel
20060: 160 unit uPSI_IBUtils;                     //VCL
20061: 161 unit uPSI_CheckLst;                    //VCL
20062: 162 unit uPSI_JvSimpleXml;                 //JCL
20063: 163 unit uPSI_JclSimpleXml;                //JCL
20064: 164 unit uPSI_JvXmlDatabase;               //JCL
20065: 165 unit uPSI_JvMaxPixel;                  //JCL
20066: 166 unit uPSI_JvItemsSearchs;              //JCL
20067: 167 unit uPSI_StExpEng2;                   //SysTools4
20068: 168 unit uPSI_StGenLog;                    //SysTools4
20069: 169 unit uPSI_JvLogFile;                   //Jcl
20070: 170 unit uPSI_CPort;                       //ComPort Lib v4.11
20071: 171 unit uPSI_CPortCtl;                    //ComPort
20072: 172 unit uPSI_CPortEsc;                    //ComPort
20073: 173 unit BarCodeScaner;                    //ComPort
20074: 174 unit uPSI_JvGraph;                     //JCL
20075: 175 unit uPSI_JvComCtrls;                  //JCL
20076: 176 unit uPSI_GUITesting;                  //D Unit
20077: 177 unit uPSI_JvFindFiles;                 //JCL
20078: 178 unit uPSI_StSystem;                    //SysTools4
20079: 179 unit uPSI_JvKeyboardStates;            //JCL
20080: 180 unit uPSI_JvMail;                      //JCL
20081: 181 unit uPSI_JclConsole;                  //JCL
20082: 182 unit uPSI_JclLANMan;                   //JCL
20083: 183 unit uPSI_IdCustomHTTPServer;          //Indy
20084: 184 unit IdHTTPServer                      //Indy
20085: 185 unit uPSI_IdTCPServer;                 //Indy
20086: 186 unit uPSI_IdSocketHandle;              //Indy
20087: 187 unit uPSI_IdIOHandlerSocket;           //Indy
20088: 188 unit IdIOHandler;                      //Indy
20089: 189 unit uPSI_cutils;                      //Bloodshed
20090: 190 unit uPSI_BoldUtils;                   //boldsoft
20091: 191 unit uPSI_IdSimpleServer;              //Indy
20092: 192 unit uPSI_IdSSLOpenSSL;                //Indy
20093: 193 unit uPSI_IdMultipartFormData;         //Indy
20094: 194 unit uPSI_SynURIOpener;                //SynEdit
20095: 195 unit uPSI_PerlRegEx;                   //PCRE
20096: 196 unit uPSI_IdHeaderList;                //Indy
20097: 197 unit uPSI_StFirst;                     //SysTools4
20098: 198 unit uPSI_JvCtrls;                     //JCL
20099: 199 unit uPSI_IdTrivialFTPBase;            //Indy
20100: 200 unit uPSI_IdTrivialFTP;                //Indy
20101: 201 unit uPSI_IdUDPBase;                   //Indy
20102: 202 unit uPSI_IdUDPClient;                 //Indy
20103: 203 unit uPSI_utypes;                      //for DMath.DLL
20104: 204 unit uPSI_ShellAPI;                    //Borland
20105: 205 unit uPSI_IdRemoteCMDClient;           //Indy
20106: 206 unit uPSI_IdRemoteCMDServer;           //Indy
20107: 207 unit IdRexecServer;                    //Indy
20108: 208 unit IdRexec; (unit uPSI_IdRexec;)     //Indy
20109: 209 unit IdUDPServer;                      //Indy
20110: 210 unit IdTimeUDPServer;                  //Indy
20111: 211 unit IdTimeServer;                     //Indy
20112: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
20113: 213 unit uPSI_IdIPWatch;                   //Indy
20114: 214 unit uPSI_IdIrcServer;                 //Indy
20115: 215 unit uPSI_IdMessageCollection;         //Indy
20116: 216 unit uPSI_cPEM;                        //Fundamentals 4
20117: 217 unit uPSI_cFundamentUtils;             //Fundamentals 4
20118: 218 unit uPSI_uwinplot;                    //DMath
20119: 219 unit uPSI_xrtl_util_CPUUtils;          //ExtentedRTL
20120: 220 unit uPSI_GR32_System;                 //Graphics32
20121: 221 unit uPSI_cFileUtils;                  //Fundamentals 4
20122: 222 unit uPSI_cDateTime; (timemachine)    //Fundamentals 4
20123: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
20124: 224 unit uPSI_cRandom;                     //Fundamentals 4
20125: 225 unit uPSI_ueval;                       //DMath
20126: 226 unit uPSI_xrtl_net_URIUtils;           //ExtendedRTL
20127: 227 unit xrtl_net_URIUtils;                //ExtendedRTL
20128: 228 unit uPSI_ufft;  (FFT)                 //DMath
20129: 229 unit uPSI_DBXChannel;                  //Delphi
20130: 230 unit uPSI_DBXIndyChannel;              //Delphi Indy
20131: 231 unit uPSI_xrtl_util_COMCat;            //ExtendedRTL
```

```
20132: 232 unit uPSI_xrtl_util_StrUtils;            //ExtendedRTL
20133: 233 unit uPSI_xrtl_util_VariantUtils;        //ExtendedRTL
20134: 234 unit uPSI_xrtl_util_FileUtils;           //ExtendedRTL
20135: 235 unit xrtl_util_Compat;                   //ExtendedRTL
20136: 236 unit uPSI_OleAuto;                       //Borland
20137: 237 unit uPSI_xrtl_util_COMUtils;            //ExtendedRTL
20138: 238 unit uPSI_CmAdmCtl;                      //Borland
20139: 239 unit uPSI_ValEdit2;                      //VCL
20140: 240 unit uPSI_GR32;  //Graphics32           //Graphics32
20141: 241 unit uPSI_GR32_Image;                    //Graphics32
20142: 242 unit uPSI_xrtl_util_TimeUtils;           //ExtendedRTL
20143: 243 unit uPSI_xrtl_util_TimeZone;            //ExtendedRTL
20144: 244 unit uPSI_xrtl_util_TimeStamp;           //ExtendedRTL
20145: 245 unit uPSI_xrtl_util_Map;                 //ExtendedRTL
20146: 246 unit uPSI_xrtl_util_Set;                 //ExtendedRTL
20147: 247 unit uPSI_CPortMonitor;                  //ComPort
20148: 248 unit uPSI_StIniStm;                      //SysTools4
20149: 249 unit uPSI_GR32_ExtImage;                 //Graphics32
20150: 250 unit uPSI_GR32_OrdinalMaps;              //Graphics32
20151: 251 unit uPSI_GR32_Rasterizers;              //Graphics32
20152: 252 unit uPSI_xrtl_util_Exception;           //ExtendedRTL
20153: 253 unit uPSI_xrtl_util_Value;               //ExtendedRTL
20154: 254 unit uPSI_xrtl_util_Compare;             //ExtendedRTL
20155: 255 unit uPSI_FlatSB;                        //VCL
20156: 256 unit uPSI_JvAnalogClock;                 //JCL
20157: 257 unit uPSI_JvAlarms;                      //JCL
20158: 258 unit uPSI_JvSQLS;                        //JCL
20159: 259 unit uPSI_JvDBSecur;                     //JCL
20160: 260 unit uPSI_JvDBQBE;                       //JCL
20161: 261 unit uPSI_JvStarfield;                   //JCL
20162: 262 unit uPSI_JVCLMiscal;                    //JCL
20163: 263 unit uPSI_JvProfiler32;                  //JCL
20164: 264 unit uPSI_JvDirectories,                 //JCL
20165: 265 unit uPSI_JclSchedule,                   //JCL
20166: 266 unit uPSI_JclSvcCtrl,                    //JCL
20167: 267 unit uPSI_JvSoundControl,                //JCL
20168: 268 unit uPSI_JvBDESQLScript,                //JCL
20169: 269 unit uPSI_JvgDigits,                     //JCL>
20170: 270 unit uPSI_ImgList;                       //TCustomImageList
20171: 271 unit uPSI_JclMIDI;                       //JCL>
20172: 272 unit uPSI_JclWinMidi;                    //JCL>
20173: 273 unit uPSI_JclNTFS;                       //JCL>
20174: 274 unit uPSI_JclAppInst;                    //JCL>
20175: 275 unit uPSI_JvRle;                         //JCL>
20176: 276 unit uPSI_JvRas32;                       //JCL>
20177: 277 unit uPSI_JvImageDrawThread,             //JCL>
20178: 278 unit uPSI_JvImageWindow,                 //JCL>
20179: 279 unit uPSI_JvTransparentForm;             //JCL>
20180: 280 unit uPSI_JvWinDialogs;                  //JCL>
20181: 281 unit uPSI_JvSimLogic,                    //JCL>
20182: 282 unit uPSI_JvSimIndicator,                //JCL>
20183: 283 unit uPSI_JvSimPID,                      //JCL>
20184: 284 unit uPSI_JvSimPIDLinker,                //JCL>
20185: 285 unit uPSI_IdRFCReply;                    //Indy
20186: 286 unit uPSI_IdIdent;                       //Indy
20187: 287 unit uPSI_IdIdentServer;                 //Indy
20188: 288 unit uPSI_JvPatchFile;                   //JCL
20189: 289 unit uPSI_StNetPfm;                      //SysTools4
20190: 290 unit uPSI_StNet;                         //SysTools4
20191: 291 unit uPSI_JclPeImage;                    //JCL
20192: 292 unit uPSI_JclPrint;                      //JCL
20193: 293 unit uPSI_JclMime;                       //JCL
20194: 294 unit uPSI_JvRichEdit;                    //JCL
20195: 295 unit uPSI_JvDBRichEd;                    //JCL
20196: 296 unit uPSI_JvDice;                        //JCL
20197: 297 unit uPSI_JvFloatEdit;                   //JCL 3.9.8
20198: 298 unit uPSI_JvDirFrm;                      //JCL
20199: 299 unit uPSI_JvDualList;                    //JCL
20200: 300 unit uPSI_JvSwitch;                      ////JCL
20201: 301 unit uPSI_JvTimerLst;                   ////JCL
20202: 302 unit uPSI_JvMemTable;                   //JCL
20203: 303 unit uPSI_JvObjStr;                     //JCL
20204: 304 unit uPSI_StLArr;                       //SysTools4
20205: 305 unit uPSI_StWmDCpy;                     //SysTools4
20206: 306 unit uPSI_StText;                       //SysTools4
20207: 307 unit uPSI_StNTLog;                      //SysTools4
20208: 308 unit uPSI_xrtl_math_Integer;            //ExtendedRTL
20209: 309 unit uPSI_JvImagPrvw;                   //JCL
20210: 310 unit uPSI_JvFormPatch;                  //JCL
20211: 311 unit uPSI_JvPicClip;                    //JCL
20212: 312 unit uPSI_JvDataConv;                   //JCL
20213: 313 unit uPSI_JvCpuUsage;                   //JCL
20214: 314 unit uPSI_JclUnitConv_mX2;              //JCL
20215: 315 unit JvDualListForm;                    //JCL
20216: 316 unit uPSI_JvCpuUsage2;                  //JCL
20217: 317 unit uPSI_JvParserForm;                 //JCL
20218: 318 unit uPSI_JvJanTreeView;                //JCL
20219: 319 unit uPSI_JvTransLED;                   //JCL
20220: 320 unit uPSI_JvPlaylist;                   //JCL
```

```
20221: 321 unit uPSI_JvFormAutoSize;                //JCL
20222: 322 unit uPSI_JvYearGridEditForm;            //JCL
20223: 323 unit uPSI_JvMarkupCommon;                //JCL
20224: 324 unit uPSI_JvChart;                       //JCL
20225: 325 unit uPSI_JvXPCore;                      //JCL
20226: 326 unit uPSI_JvXPCoreUtils;                 //JCL
20227: 327 unit uPSI_StatsClasses;                  //mX4
20228: 328 unit uPSI_ExtCtrls2;                     //VCL
20229: 329 unit uPSI_JvUrlGrabbers;                 //JCL
20230: 330 unit uPSI_JvXmlTree;                     //JCL
20231: 331 unit uPSI_JvWavePlayer;                  //JCL
20232: 332 unit uPSI_JvUnicodeCanvas;               //JCL
20233: 333 unit uPSI_JvTFUtils;                     //JCL
20234: 334 unit uPSI_IdServerIOHandler;             //Indy
20235: 335 unit uPSI_IdServerIOHandlerSocket;       //Indy
20236: 336 unit uPSI_IdMessageCoder;                //Indy
20237: 337 unit uPSI_IdMessageCoderMIME;            //Indy
20238: 338 unit uPSI_IdMIMETypes;                   //Indy
20239: 339 unit uPSI_JvConverter;                   //JCL
20240: 340 unit uPSI_JvCsvParse;                    //JCL
20241: 341 unit uPSI_umath;  unit uPSI_ugamma;      //DMath
20242: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
20243: 343 unit uPSI_JvDBGridExport;                //JCL
20244: 344 unit uPSI_JvgExport;                     //JCL
20245: 345 unit uPSI_JvSerialMaker;                 //JCL
20246: 346 unit uPSI_JvWin32;                       //JCL
20247: 347 unit uPSI_JvPaintFX;                     //JCL
20248: 348 unit uPSI_JvOracleDataSet; (beta)        //JCL
20249: 349 unit uPSI_JvValidators; (preview)        //JCL
20250: 350 unit uPSI_JvNTEventLog;                  //JCL
20251: 351 unit uPSI_ShellZipTool;                  //mX4
20252: 352 unit uPSI_JvJoystick;                    //JCL
20253: 353 unit uPSI_JvMailSlots;                   //JCL
20254: 354 unit uPSI_JclComplex;                    //JCL
20255: 355 unit uPSI_SynPdf;                        //Synopse
20256: 356 unit uPSI_Registry;                      //VCL
20257: 357 unit uPSI_TlHelp32;                      //VCL
20258: 358 unit uPSI_JclRegistry;                   //JCL
20259: 359 unit uPSI_JvAirBrush;                    //JCL
20260: 360 unit uPSI_mORMotReport;                  //Synopse
20261: 361 unit uPSI_JclLocales;                    //JCL
20262: 362 unit uPSI_SynEdit;                       //SynEdit
20263: 363 unit uPSI_SynEditTypes;                  //SynEdit
20264: 364 unit uPSI_SynMacroRecorder;              //SynEdit
20265: 365 unit uPSI_LongIntList;                   //SynEdit
20266: 366 unit uPSI_devcutils;                     //DevC
20267: 367 unit uPSI_SynEditMiscClasses;            //SynEdit
20268: 368 unit uPSI_SynEditRegexSearch;            //SynEdit
20269: 369 unit uPSI_SynEditHighlighter;            //SynEdit
20270: 370 unit uPSI_SynHighlighterPas;             //SynEdit
20271: 371 unit uPSI_JvSearchFiles;                 //JCL
20272: 372 unit uPSI_SynHighlighterAny;             //Lazarus
20273: 373 unit uPSI_SynEditKeyCmds;                //SynEdit
20274: 374 unit uPSI_SynEditMiscProcs,              //SynEdit
20275: 375 unit uPSI_SynEditKbdHandler              //SynEdit
20276: 376 unit uPSI_JvAppInst,                     //JCL
20277: 377 unit uPSI_JvAppEvent;                    //JCL
20278: 378 unit uPSI_JvAppCommand;                  //JCL
20279: 379 unit uPSI_JvAnimTitle;                   //JCL
20280: 380 unit uPSI_JvAnimatedImage;               //JCL
20281: 381 unit uPSI_SynEditExport;                 //SynEdit
20282: 382 unit uPSI_SynExportHTML;                 //SynEdit
20283: 383 unit uPSI_SynExportRTF;                  //SynEdit
20284: 384 unit uPSI_SynEditSearch;                 //SynEdit
20285: 385 unit uPSI_fMain_back                     //maXbox;
20286: 386 unit uPSI_JvZoom;                        //JCL
20287: 387 unit uPSI_PMrand;                        //PM
20288: 388 unit uPSI_JvSticker;                     //JCL
20289: 389 unit uPSI_XmlVerySimple;                 //mX4
20290: 390 unit uPSI_Services;                      //ExtPascal
20291: 391 unit uPSI_ExtPascalUtils;                //ExtPascal
20292: 392 unit uPSI_SocketsDelphi;                 //ExtPascal
20293: 393 unit uPSI_StBarC;                        //SysTools
20294: 394 unit uPSI_StDbBarC;                      //SysTools
20295: 395 unit uPSI_StBarPN;                       //SysTools
20296: 396 unit uPSI_StDbPNBC;                      //SysTools
20297: 397 unit uPSI_StDb2DBC;                      //SysTools
20298: 398 unit uPSI_StMoney;                       //SysTools
20299: 399 unit uPSI_JvForth;                       //JCL
20300: 400 unit uPSI_RestRequest;                   //mX4
20301: 401 unit uPSI_HttpRESTConnectionIndy;        //mX4
20302: 402 unit uPSI_JvXmlDatabase; //update       //JCL
20303: 403 unit uPSI_StAstro;                       //SysTools
20304: 404 unit uPSI_StSort;                        //SysTools
20305: 405 unit uPSI_StDate;                        //SysTools
20306: 406 unit uPSI_StDateSt;                      //SysTools
20307: 407 unit uPSI_StBase;                        //SysTools
20308: 408 unit uPSI_StVInfo;                       //SysTools
20309: 409 unit uPSI_JvBrowseFolder;                //JCL
```

```
20310: 410 unit uPSI_JvBoxProcs;                    //JCL
20311: 411 unit uPSI_urandom; (unit uranuvag;)      //DMath
20312: 412 unit uPSI_usimann; (unit ugenalg;)       //DMath
20313: 413 unit uPSI_JvHighlighter;                 //JCL
20314: 414 unit uPSI_Diff;                          //mX4
20315: 415 unit uPSI_SpringWinAPI;                  //DSpring
20316: 416 unit uPSI_StBits;                        //SysTools
20317: 417 unit uPSI_TomDBQue;                      //mX4
20318: 418 unit uPSI_MultilangTranslator;           //mX4
20319: 419 unit uPSI_HyperLabel;                    //mX4
20320: 420 unit uPSI_Starter;                       //mX4
20321: 421 unit uPSI_FileAssocs;                    //devC
20322: 422 unit uPSI_devFileMonitorX;               //devC
20323: 423 unit uPSI_devrun;                        //devC
20324: 424 unit uPSI_devExec;                       //devC
20325: 425 unit uPSI_oysUtils;                      //devC
20326: 426 unit uPSI_DosCommand;                    //devC
20327: 427 unit uPSI_CppTokenizer;                  //devC
20328: 428 unit uPSI_JvHLParser;                    //devC
20329: 429 unit uPSI_JclMapi;                       //JCL
20330: 430 unit uPSI_JclShell;                      //JCL
20331: 431 unit uPSI_JclCOM;                        //JCL
20332: 432 unit uPSI_GR32_Math;                     //Graphics32
20333: 433 unit uPSI_GR32_LowLevel;                 //Graphics32
20334: 434 unit uPSI_SimpleHl;                      //mX4
20335: 435 unit uPSI_GR32_Filters,                  //Graphics32
20336: 436 unit uPSI_GR32_VectorMaps;               //Graphics32
20337: 437 unit uPSI_cXMLFunctions;                 //Fundamentals 4
20338: 438 unit uPSI_JvTimer;                       //JCL
20339: 439 unit uPSI_cHTTPUtils;                    //Fundamentals 4
20340: 440 unit uPSI_cTLSUtils;                     //Fundamentals 4
20341: 441 unit uPSI_JclGraphics;                   //JCL
20342: 442 unit uPSI_JclSynch;                      //JCL
20343: 443 unit uPSI_IdTelnet;                      //Indy
20344: 444 unit uPSI_IdTelnetServer,                //Indy
20345: 445 unit uPSI_IdEcho,                        //Indy
20346: 446 unit uPSI_IdEchoServer,                  //Indy
20347: 447 unit uPSI_IdEchoUDP,                     //Indy
20348: 448 unit uPSI_IdEchoUDPServer,               //Indy
20349: 449 unit uPSI_IdSocks,                       //Indy
20350: 450 unit uPSI_IdAntiFreezeBase;             //Indy
20351: 451 unit uPSI_IdHostnameServer;             //Indy
20352: 452 unit uPSI_IdTunnelCommon,                //Indy
20353: 453 unit uPSI_IdTunnelMaster,                //Indy
20354: 454 unit uPSI_IdTunnelSlave,                 //Indy
20355: 455 unit uPSI_IdRSH,                         //Indy
20356: 456 unit uPSI_IdRSHServer,                   //Indy
20357: 457 unit uPSI_Spring_Cryptography_Utils;     //Spring4Delphi
20358: 458 unit uPSI_MapReader,                     //devC
20359: 459 unit uPSI_LibTar,                        //devC
20360: 460 unit uPSI_IdStack;                       //Indy
20361: 461 unit uPSI_IdBlockCipherIntercept;        //Indy
20362: 462 unit uPSI_IdChargenServer;               //Indy
20363: 463 unit uPSI_IdFTPServer,                   //Indy
20364: 464 unit uPSI_IdException,                   //Indy
20365: 465 unit uPSI_utexplot;                      //DMath
20366: 466 unit uPSI_uwinstr;                       //DMath
20367: 467 unit uPSI_VarRecUtils;                   //devC
20368: 468 unit uPSI_JvStringListToHtml,            //JCL
20369: 469 unit uPSI_JvStringHolder,                //JCL
20370: 470 unit uPSI_IdCoder;                       //Indy
20371: 471 unit uPSI_SynHighlighterDfm;             //Synedit
20372: 472 unit uHighlighterProcs; in 471           //Synedit
20373: 473 unit uPSI_LazFileUtils,                  //LCL
20374: 474 unit uPSI_IDECmdLine;                    //LCL
20375: 475 unit uPSI_lazMasks;                      //LCL
20376: 476 unit uPSI_ip_misc;                       //mX4
20377: 477 unit uPSI_Barcode;                       //LCL
20378: 478 unit uPSI_SimpleXML;                     //LCL
20379: 479 unit uPSI_JclIniFiles;                   //JCL
20380: 480 unit uPSI_D2XXUnit;  {$X-}               //FTDI
20381: 481 unit uPSI_JclDateTime;                   //JCL
20382: 482 unit uPSI_JclEDI;                        //JCL
20383: 483 unit uPSI_JclMiscel2;                    //JCL
20384: 484 unit uPSI_JclValidation;                 //JCL
20385: 485 unit uPSI_JclAnsiStrings; {-PString}     //JCL
20386: 486 unit uPSI_SynEditMiscProcs2;             //Synedit
20387: 487 unit uPSI_JclStreams;                    //JCL
20388: 488 unit uPSI_QRCode;                        //mX4
20389: 489 unit uPSI_BlockSocket;                   //ExtPascal
20390: 490 unit uPSI_Masks,Utils                    //VCL
20391: 491 unit uPSI_synautil;                      //Synapse!
20392: 492 unit uPSI_JclMath_Class;                 //JCL RTL
20393: 493 unit ugamdist; //Gamma function         //DMath
20394: 494 unit uibeta, ucorrel; //IBeta           //DMath
20395: 495 unit uPSI_SRMgr;                         //mX4
20396: 496 unit uPSI_HotLog;                        //mX4
20397: 497 unit uPSI_DebugBox;                      //mX4
20398: 498 unit uPSI_ustrings;                      //DMath
```

```
20399: 499 unit uPSI_uregtest;                    //DMath
20400: 500 unit uPSI_usimplex;                    //DMath
20401: 501 unit uPSI_uhyper;                      //DMath
20402: 502 unit uPSI_IdHL7;                       //Indy
20403: 503 unit uPSI_IdIPMCastBase,               //Indy
20404: 504 unit uPSI_IdIPMCastServer;             //Indy
20405: 505 unit uPSI_IdIPMCastClient;             //Indy
20406: 506 unit uPSI_unlfit; //nlregression      //DMath
20407: 507 unit uPSI_IdRawHeaders;                //Indy
20408: 508 unit uPSI_IdRawClient;                 //Indy
20409: 509 unit uPSI_IdRawFunctions;              //Indy
20410: 510 unit uPSI_IdTCPStream;                 //Indy
20411: 511 unit uPSI_IdSNPP;                      //Indy
20412: 512 unit uPSI_St2DBarC;                    //SysTools
20413: 513 unit uPSI_ImageWin;  //FTL             //VCL
20414: 514 unit uPSI_CustomDrawTreeView; //FTL    //VCL
20415: 515 unit uPSI_GraphWin;  //FTL             //VCL
20416: 516 unit uPSI_actionMain;  //FTL           //VCL
20417: 517 unit uPSI_StSpawn;                     //SysTools
20418: 518 unit uPSI_CtlPanel;                    //VCL
20419: 519 unit uPSI_IdLPR;                       //Indy
20420: 520 unit uPSI_SockRequestInterpreter;      //Indy
20421: 521 unit uPSI_ulambert;                    //DMath
20422: 522 unit uPSI_ucholesk;                    //DMath
20423: 523 unit uPSI_SimpleDS;                    //VCL
20424: 524 unit uPSI_DBXSqlScanner;               //VCL
20425: 525 unit uPSI_DBXMetaDataUtil;             //VCL
20426: 526 unit uPSI_Chart;                       //TEE
20427: 527 unit uPSI_TeeProcs;                    //TEE
20428: 528 unit mXBDEUtils;                       //mX4
20429: 529 unit uPSI_MDIEdit;                     //VCL
20430: 530 unit uPSI_CopyPrsr;                    //VCL
20431: 531 unit uPSI_SockApp;                     //VCL
20432: 532 unit uPSI_AppEvnts;                    //VCL
20433: 533 unit uPSI_ExtActns;                    //VCL
20434: 534 unit uPSI_TeEngine;                    //TEE
20435: 535 unit uPSI_CoolMain; //browser          //VCL
20436: 536 unit uPSI_StCRC;                       //SysTools
20437: 537 unit uPSI_StDecMth2;                   //SysTools
20438: 538 unit uPSI_frmExportMain;               //Synedit
20439: 539 unit uPSI_SynDBEdit;                   //Synedit
20440: 540 unit uPSI_SynEditWildcardSearch;       //Synedit
20441: 541 unit uPSI_BoldComUtils;                //BOLD
20442: 542 unit uPSI_BoldIsoDateTime;             //BOLD
20443: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils  //BOLD
20444: 544 unit uPSI_BoldXMLRequests;             //BOLD
20445: 545 unit uPSI_BoldStringList;              //BOLD
20446: 546 unit uPSI_BoldFileHandler;             //BOLD
20447: 547 unit uPSI_BoldContainers;              //BOLD
20448: 548 unit uPSI_BoldQueryUserDlg;            //BOLD
20449: 549 unit uPSI_BoldWinINet;                 //BOLD
20450: 550 unit uPSI_BoldQueue;                   //BOLD
20451: 551 unit uPSI_JvPcx;                       //JCL
20452: 552 unit uPSI_IdWhois;                     //Indy
20453: 553 unit uPSI_IdWhoIsServer;               //Indy
20454: 554 unit uPSI_IdGopher;                    //Indy
20455: 555 unit uPSI_IdDateTimeStamp;             //Indy
20456: 556 unit uPSI_IdDayTimeServer;             //Indy
20457: 557 unit uPSI_IdDayTimeUDP;                //Indy
20458: 558 unit uPSI_IdDayTimeUDPServer;          //Indy
20459: 559 unit uPSI_IdDICTServer;                //Indy
20460: 560 unit uPSI_IdDiscardServer;             //Indy
20461: 561 unit uPSI_IdDiscardUDPServer;          //Indy
20462: 562 unit uPSI_IdMappedFTP;                 //Indy
20463: 563 unit uPSI_IdMappedPortTCP;             //Indy
20464: 564 unit uPSI_IdGopherServer;              //Indy
20465: 565 unit uPSI_IdQotdServer;                //Indy
20466: 566 unit uPSI_JvRgbToHtml;                 //JCL
20467: 567 unit uPSI_JvRemLog,                    //JCL
20468: 568 unit uPSI_JvSysComp;                   //JCL
20469: 569 unit uPSI_JvTMTL;                      //JCL
20470: 570 unit uPSI_JvWinampAPI;                 //JCL
20471: 571 unit uPSI_MSysUtils;                   //mX4
20472: 572 unit uPSI_ESBMaths;                    //ESB
20473: 573 unit uPSI_ESBMaths2;                   //ESB
20474: 574 unit uPSI_uLkJSON;                     //Lk
20475: 575 unit uPSI_ZURL;  //Zeos                //Zeos
20476: 576 unit uPSI_ZSysUtils;                   //Zeos
20477: 577 unit unaUtils internals                //UNA
20478: 578 unit uPSI_ZMatchPattern;               //Zeos
20479: 579 unit uPSI_ZClasses;                    //Zeos
20480: 580 unit uPSI_ZCollections;                //Zeos
20481: 581 unit uPSI_ZEncoding;                   //Zeos
20482: 582 unit uPSI_IdRawBase;                   //Indy
20483: 583 unit uPSI_IdNTLM;                      //Indy
20484: 584 unit uPSI_IdNNTP;                      //Indy
20485: 585 unit uPSI_usniffer; //PortScanForm     //mX4
20486: 586 unit uPSI_IdCoderMIME;                 //Indy
20487: 587 unit uPSI_IdCoderUUE;                  //Indy
```

```
20488: 588 unit uPSI_IdCoderXXE;                    //Indy
20489: 589 unit uPSI_IdCoder3to4;                   //Indy
20490: 590 unit uPSI_IdCookie;                      //Indy
20491: 591 unit uPSI_IdCookieManager;               //Indy
20492: 592 unit uPSI_WDosSocketUtils;               //WDos
20493: 593 unit uPSI_WDosPlcUtils;                  //WDos
20494: 594 unit uPSI_WDosPorts;                     //WDos
20495: 595 unit uPSI_WDosResolvers;                 //WDos
20496: 596 unit uPSI_WDosTimers;                    //WDos
20497: 597 unit uPSI_WDosPlcs;                      //WDos
20498: 598 unit uPSI_WDosPneumatics;                //WDos
20499: 599 unit uPSI_IdFingerServer;                //Indy
20500: 600 unit uPSI_IdDNSResolver;                 //Indy
20501: 601 unit uPSI_IdHTTPWebBrokerBridge;         //Indy
20502: 602 unit uPSI_IdIntercept;                   //Indy
20503: 603 unit uPSI_IdIPMCastBase;                 //Indy
20504: 604 unit uPSI_IdLogBase;                     //Indy
20505: 605 unit uPSI_IdIOHandlerStream;             //Indy
20506: 606 unit uPSI_IdMappedPortUDP;               //Indy
20507: 607 unit uPSI_IdQOTDUDPServer;               //Indy
20508: 608 unit uPSI_IdQOTDUDP;                     //Indy
20509: 609 unit uPSI_IdSysLog;                      //Indy
20510: 610 unit uPSI_IdSysLogServer;                //Indy
20511: 611 unit uPSI_IdSysLogMessage;               //Indy
20512: 612 unit uPSI_IdTimeServer;                  //Indy
20513: 613 unit uPSI_IdTimeUDP;                     //Indy
20514: 614 unit uPSI_IdTimeUDPServer;               //Indy
20515: 615 unit uPSI_IdUserAccounts;                //Indy
20516: 616 unit uPSI_TextUtils;                     //mX4
20517: 617 unit uPSI_MandelbrotEngine;              //mX4
20518: 618 unit uPSI_delphi_arduino_Unit1;          //mX4
20519: 619 unit uPSI_DTDSchema2;                    //mX4
20520: 620 unit uPSI_fplotMain;                     //DMath
20521: 621 unit uPSI_FindFileIter;                  //mX4
20522: 622 unit uPSI_PppState;  (JclStrHashMap)     //PPP
20523: 623 unit uPSI_PppParser;                     //PPP
20524: 624 unit uPSI_PppLexer;                      //PPP
20525: 625 unit uPSI_PCharUtils;                    //PPP
20526: 626 unit uPSI_uJSON;                         //WU
20527: 627 unit uPSI_JclStrHashMap;                 //JCL
20528: 628 unit uPSI_JclHookExcept;                 //JCL
20529: 629 unit uPSI_EncdDecd;                      //VCL
20530: 630 unit uPSI_SockAppReg;                    //VCL
20531: 631 unit uPSI_PJFileHandle;                  //PJ
20532: 632 unit uPSI_PJEnvVars;                     //PJ
20533: 633 unit uPSI_PJPipe;                        //PJ
20534: 634 unit uPSI_PJPipeFilters;                 //PJ
20535: 635 unit uPSI_PJConsoleApp;                  //PJ
20536: 636 unit uPSI_UConsoleAppEx;                 //PJ
20537: 637 unit uPSI_DbxSocketChannelNative,        //VCL
20538: 638 unit uPSI_DbxDataGenerator,              //VCL
20539: 639 unit uPSI_DBXClient;                     //VCL
20540: 640 unit uPSI_IdLogEvent;                    //Indy
20541: 641 unit uPSI_Reversi;                       //mX4
20542: 642 unit uPSI_Geometry;                      //mX4
20543: 643 unit uPSI_IdSMTPServer;                  //Indy
20544: 644 unit uPSI_Textures;                      //mX4
20545: 645 unit uPSI_IBX;                           //VCL
20546: 646 unit uPSI_IWDBCommon;                    //VCL
20547: 647 unit uPSI_SortGrid;                      //mX4
20548: 648 unit uPSI_IB;                            //VCL
20549: 649 unit uPSI_IBScript;                      //VCL
20550: 650 unit uPSI_JvCSVBaseControls;             //JCL
20551: 651 unit uPSI_Jvg3DColors;                   //JCL
20552: 652 unit uPSI_JvHLEditor;  //beat            //JCL
20553: 653 unit uPSI_JvShellHook;                   //JCL
20554: 654 unit uPSI_DBCommon2                      //VCL
20555: 655 unit uPSI_JvSHFileOperation;             //JCL
20556: 656 unit uPSI_uFilexport;                    //mX4
20557: 657 unit uPSI_JvDialogs;                     //JCL
20558: 658 unit uPSI_JvDBTreeView;                  //JCL
20559: 659 unit uPSI_JvDBUltimGrid;                 //JCL
20560: 660 unit uPSI_JvDBQueryParamsForm;           //JCL
20561: 661 unit uPSI_JvExControls;                  //JCL
20562: 662 unit uPSI_JvBDEMemTable;                 //JCL
20563: 663 unit uPSI_JvCommStatus;                  //JCL
20564: 664 unit uPSI_JvMailSlots2;                  //JCL
20565: 665 unit uPSI_JvgWinMask;                    //JCL
20566: 666 unit uPSI_StEclpse;                      //SysTools
20567: 667 unit uPSI_StMime;                        //SysTools
20568: 668 unit uPSI_StList;                        //SysTools
20569: 669 unit uPSI_StMerge;                       //SysTools
20570: 670 unit uPSI_StStrS;                        //SysTools
20571: 671 unit uPSI_StTree,                        //SysTools
20572: 672 unit uPSI_StVArr;                        //SysTools
20573: 673 unit uPSI_StRegIni;                      //SysTools
20574: 674 unit uPSI_urkf;                          //DMath
20575: 675 unit uPSI_usvd;                          //DMath
20576: 676 unit uPSI_DepWalkUtils;                  //JCL
```

```
20577: 677 unit uPSI_OptionsFrm;                    //JCL
20578: 678 unit yuvconverts;                        //mX4
20579: 679 uPSI_JvPropAutoSave;                     //JCL
20580: 680 uPSI_AclAPI;                             //alcinoe
20581: 681 uPSI_AviCap;                             //alcinoe
20582: 682 uPSI_ALAVLBinaryTree;                    //alcinoe
20583: 683 uPSI_ALFcnMisc;                          //alcinoe
20584: 684 uPSI_ALStringList;                       //alcinoe
20585: 685 uPSI_ALQuickSortList;                    //alcinoe
20586: 686 uPSI_ALStaticText;                       //alcinoe
20587: 687 uPSI_ALJSONDoc;                          //alcinoe
20588: 688 uPSI_ALGSMComm;                          //alcinoe
20589: 689 uPSI_ALWindows;                          //alcinoe
20590: 690 uPSI_ALMultiPartFormDataParser;          //alcinoe
20591: 691 uPSI_ALHttpCommon;                       //alcinoe
20592: 692 uPSI_ALWebSpider,                        //alcinoe
20593: 693 uPSI_ALHttpClient;                       //alcinoe
20594: 694 uPSI_ALFcnHTML;                          //alcinoe
20595: 695 uPSI_ALFTPClient;                        //alcinoe
20596: 696 uPSI_ALInternetMessageCommon;            //alcinoe
20597: 697 uPSI_ALWininetHttpClient;                //alcinoe
20598: 698 uPSI_ALWinInetFTPClient;                 //alcinoe
20599: 699 uPSI_ALWinHttpWrapper;                   //alcinoe
20600: 700 uPSI_ALWinHttpClient;                    //alcinoe
20601: 701 uPSI_ALFcnWinSock;                       //alcinoe
20602: 702 uPSI_ALFcnSQL;                           //alcinoe
20603: 703 uPSI_ALFcnCGI;                           //alcinoe
20604: 704 uPSI_ALFcnExecute;                       //alcinoe
20605: 705 uPSI_ALFcnFile;                          //alcinoe
20606: 706 uPSI_ALFcnMime;                          //alcinoe
20607: 707 uPSI_ALPhpRunner;                        //alcinoe
20608: 708 uPSI_ALGraphic;                          //alcinoe
20609: 709 uPSI_ALIniFiles;                         //alcinoe
20610: 710 uPSI_ALMemCachedClient;                  //alcinoe
20611: 711 unit uPSI_MyGrids;                       //mX4
20612: 712 uPSI_ALMultiPartMixedParser              //alcinoe
20613: 713 uPSI_ALSMTPClient                        //alcinoe
20614: 714 uPSI_ALNNTPClient;                       //alcinoe
20615: 715 uPSI_ALHintBalloon;                      //alcinoe
20616: 716 unit uPSI_ALXmlDoc;                      //alcinoe
20617: 717 unit uPSI_IPCThrd;                       //VCL
20618: 718 unit uPSI_MonForm;                       //VCL
20619: 719 unit uPSI_TeCanvas;                      //Orpheus
20620: 720 unit uPSI_Ovcmisc;                       //Orpheus
20621: 721 unit uPSI_ovcfiler;                      //Orpheus
20622: 722 unit uPSI_ovcstate;                      //Orpheus
20623: 723 unit uPSI_ovccoco;                       //Orpheus
20624: 724 unit uPSI_ovcrvexp;                      //Orpheus
20625: 725 unit uPSI_OvcFormatSettings;             //Orpheus
20626: 726 unit uPSI_OvcUtils;                      //Orpheus
20627: 727 unit uPSI_ovcstore;                      //Orpheus
20628: 728 unit uPSI_ovcstr;                        //Orpheus
20629: 729 unit uPSI_ovcmru;                        //Orpheus
20630: 730 unit uPSI_ovccmd;                        //Orpheus
20631: 731 unit uPSI_ovctimer;                      //Orpheus
20632: 732 unit uPSI_ovcintl;                       //Orpheus
20633: 733 uPSI_AfCircularBuffer;                   //AsyncFree
20634: 734 uPSI_AfUtils;                           //AsyncFree
20635: 735 uPSI_AfSafeSync;                        //AsyncFree
20636: 736 uPSI_AfComPortCore;                     //AsyncFree
20637: 737 uPSI_AfComPort;                         //AsyncFree
20638: 738 uPSI_AfPortControls;                    //AsyncFree
20639: 739 uPSI_AfDataDispatcher;                  //AsyncFree
20640: 740 uPSI_AfViewers;                         //AsyncFree
20641: 741 uPSI_AfDataTerminal;                    //AsyncFree
20642: 742 uPSI_SimplePortMain;                    //AsyncFree
20643: 743 unit uPSI_ovcclock;                      //Orpheus
20644: 744 unit uPSI_o32intlst;                     //Orpheus
20645: 745 unit uPSI_o32ledlabel;                   //Orpheus
20646: 746 unit uPSI_AlMySqlClient;                 //alcinoe
20647: 747 unit uPSI_ALFBXClient;                   //alcinoe
20648: 748 unit uPSI_ALFcnSQL;                      //alcinoe
20649: 749 unit uPSI_AsyncTimer;                    //mX4
20650: 750 unit uPSI_ApplicationFileIO;             //mX4
20651: 751 unit uPSI_PsAPI;                         //VCLé
20652: 752 uPSI_ovcuser;                           //Orpheus
20653: 753 uPSI_ovcurl;                            //Orpheus
20654: 754 uPSI_ovcvlb;                            //Orpheus
20655: 755 uPSI_ovccolor;                          //Orpheus
20656: 756 uPSI_ALFBXLib,                          //alcinoe
20657: 757 uPSI_ovcmeter;                          //Orpheus
20658: 758 uPSI_ovcpeakm;                          //Orpheus
20659: 759 uPSI_O32BGSty;                          //Orpheus
20660: 760 uPSI_ovcBidi;                           //Orpheus
20661: 761 uPSI_ovctcary;                          //Orpheus
20662: 762 uPSI_DXPUtils;                          //mX4
20663: 763 uPSI_ALMultiPartBaseParser;             //alcinoe
20664: 764 uPSI_ALMultiPartAlternativeParser;      //alcinoe
20665: 765 uPSI_ALPOP3Client;                      //alcinoe
```

```
20666: 766 uPSI_SmallUtils;                    //mX4
20667: 767 uPSI_MakeApp;                       //mX4
20668: 768 uPSI_O32MouseMon;                   //Orpheus
20669: 769 uPSI_OvcCache;                      //Orpheus
20670: 770 uPSI_ovccalc;                       //Orpheus
20671: 771 uPSI_Joystick,                      //OpenGL
20672: 772 uPSI_ScreenSaver;                   //OpenGL
20673: 773 uPSI_XCollection,                   //OpenGL
20674: 774 uPSI_Polynomials,                   //OpenGL
20675: 775 uPSI_PersistentClasses, //9.86     //OpenGL
20676: 776 uPSI_VectorLists;                   //OpenGL
20677: 777 uPSI_XOpenGL,                      //OpenGL
20678: 778 uPSI_MeshUtils;                    //OpenGL
20679: 779 unit uPSI_JclSysUtils;             //JCL
20680: 780 unit uPSI_JclBorlandTools;         //JCL
20681: 781 unit JclFileUtils_max;             //JCL
20682: 782 uPSI_AfDataControls,               //AsyncFree
20683: 783 uPSI_GLSilhouette;                 //OpenGL
20684: 784 uPSI_JclSysUtils_class;            //JCL
20685: 785 uPSI_JclFileUtils_class;           //JCL
20686: 786 uPSI_FileUtil;                     //JCL
20687: 787 uPSI_changefind;                   //mX4
20688: 788 uPSI_cmdIntf;                      //mX4
20689: 789 uPSI_fservice;                     //mX4
20690: 790 uPSI_Keyboard;                     //OpenGL
20691: 791 uPSI_VRMLParser,                   //OpenGL
20692: 792 uPSI_GLFileVRML,                   //OpenGL
20693: 793 uPSI_Octree;                       //OpenGL
20694: 794 uPSI_GLPolyhedron,                 //OpenGL
20695: 795 uPSI_GLCrossPlatform;              //OpenGL
20696: 796 uPSI_GLParticles;                  //OpenGL
20697: 797 uPSI_GLNavigator;                  //OpenGL
20698: 798 uPSI_GLStarRecord;                 //OpenGL
20699: 799 uPSI_GLTextureCombiners;           //OpenGL
20700: 800 uPSI_GLCanvas;                     //OpenGL
20701: 801 uPSI_GeometryBB;                   //OpenGL
20702: 802 uPSI_GeometryCoordinates;          //OpenGL
20703: 803 uPSI_VectorGeometry;               //OpenGL
20704: 804 uPSI_BumpMapping;                  //OpenGL
20705: 805 uPSI_TGA;                          //OpenGL
20706: 806 uPSI_GLVectorFileObjects;          //OpenGL
20707: 807 uPSI_IMM;                          //VCL
20708: 808 uPSI_CategoryButtons;              //VCL
20709: 809 uPSI_ButtonGroup;                  //VCL
20710: 810 uPSI_DbExcept;                     //VCL
20711: 811 uPSI_AxCtrls;                      //VCL
20712: 812 uPSI_GL_actorUnit1;                //OpenGL
20713: 813 uPSI_StdVCL;                       //VCL
20714: 814 unit CurvesAndSurfaces;            //OpenGL
20715: 815 uPSI_DataAwareMain;                //AsyncFree
20716: 816 uPSI_TabNotBk;                     //VCL
20717: 817 uPSI_udwsfiler;                    //mX4
20718: 818 uPSI_synaip;                       //Synapse!
20719: 819 uPSI_synacode;                     //Synapse
20720: 820 uPSI_synachar;                     //Synapse
20721: 821 uPSI_synamisc;                     //Synapse
20722: 822 uPSI_synaser;                      //Synapse
20723: 823 uPSI_synaicnv;                     //Synapse
20724: 824 uPSI_tlntsend;                     //Synapse
20725: 825 uPSI_pingsend;                     //Synapse
20726: 826 uPSI_blcksock;                     //Synapse
20727: 827 uPSI_asn1util;                     //Synapse
20728: 828 uPSI_dnssend;                      //Synapse
20729: 829 uPSI_clamsend;                     //Synapse
20730: 830 uPSI_ldapsend;                     //Synapse
20731: 831 uPSI_mimemess;                     //Synapse
20732: 832 uPSI_slogsend;                     //Synapse
20733: 833 uPSI_mimepart;                     //Synapse
20734: 834 uPSI_mimeinln;                     //Synapse
20735: 835 uPSI_ftpsend,                      //Synapse
20736: 836 uPSI_ftptsend;                     //Synapse
20737: 837 uPSI_httpsend;                     //Synapse
20738: 838 uPSI_sntpsend;                     //Synapse
20739: 839 uPSI_smtpsend;                     //Synapse
20740: 840 uPSI_snmpsend;                     //Synapse
20741: 841 uPSI_imapsend;                     //Synapse
20742: 842 uPSI_pop3send;                     //Synapse
20743: 843 uPSI_nntpsend;                     //Synapse
20744: 844 uPSI_ssl_cryptlib;                 //Synapse
20745: 845 uPSI_ssl_openssl;                  //Synapse
20746: 846 uPSI_synhttp_daemon;               //Synapse
20747: 847 uPSI_NetWork;                      //mX4
20748: 848 uPSI_PingThread;                   //Synapse
20749: 849 uPSI_JvThreadTimer;                //JCL
20750: 850 unit uPSI_wwSystem;                //InfoPower
20751: 851 unit uPSI_IdComponent;             //Indy
20752: 852 unit uPSI_IdIOHandlerThrottle;     //Indy
20753: 853 unit uPSI_Themes;                  //VCL
20754: 854 unit uPSI_StdStyleActnCtrls;       //VCL
```

```
20755: 855 unit uPSI_UDDIHelper;                    //VCL
20756: 856 unit uPSI_IdIMAP4Server;                 //Indy
20757: 857 uPSI_VariantSymbolTable,                 //VCL //3.9.9.92
20758: 858 uPSI_udf_glob,                           //mX4
20759: 859 uPSI_TabGrid,                            //VCL
20760: 860 uPSI_JsDBTreeView,                       //mX4
20761: 861 uPSI_JsSendMail,                        //mX4
20762: 862 uPSI_dbTvRecordList,                    //mX4
20763: 863 uPSI_TreeVwEx,                          //mX4
20764: 864 uPSI_ECDataLink,                        //mX4
20765: 865 uPSI_dbTree,                           //mX4
20766: 866 uPSI_dbTreeCBox,                       //mX4
20767: 867 unit uPSI_Debug; //TfrmDebug           //mX4
20768: 868 uPSI_TimeFncs;                         //mX4
20769: 869 uPSI_FileIntf,                         //VCL
20770: 870 uPSI_SockTransport,                    //RTL
20771: 871 unit uPSI_WinInet;                     //RTL
20772: 872 unit uPSI_Wwstr;                       //mX4
20773: 873 uPSI_DBLookup,                         //VCL
20774: 874 uPSI_Hotspot,                          //mX4
20775: 875 uPSI_HList; //History List             //mX4
20776: 876 unit uPSI_DrTable;                     //VCL
20777: 877 uPSI_TConnect,                         //VCL
20778: 978 uPSI_DataBkr,                          //VCL
20779: 979 uPSI_HTTPIntr;                        //VCL
20780: 980 unit uPSI_Mathbox;                     //mX4
20781: 881 uPSI_cyIndy,                          //cY
20782: 882 uPSI_cySysUtils,                      //cY
20783: 883 uPSI_cyWinUtils,                      //cY
20784: 884 uPSI_cyStrUtils,                      //cY
20785: 885 uPSI_cyObjUtils,                      //cY
20786: 886 uPSI_cyDateUtils,                     //cY
20787: 887 uPSI_cyBDE,                           //cY
20788: 888 uPSI_cyClasses,                       //cY
20789: 889 uPSI_cyGraphics,  //3.9.9.94_2        //cY
20790: 890 unit uPSI_cyTypes;                     //cY
20791: 891 uPSI_JvDateTimePicker,                //JCL
20792: 892 uPSI_JvCreateProcess,                 //JCL
20793: 893 uPSI_JvEasterEgg,                     //JCL
20794: 894 uPSI_WinSvc,  //3.9.9.94_3            //VCL
20795: 895 uPSI_SvcMgr                           //VCL
20796: 896 unit uPSI_JvPickDate;                 //JCL
20797: 897 unit uPSI_JvNotify;                   //JCL
20798: 898 uPSI_JvStrHlder                       //JCL
20799: 899 unit uPSI_JclNTFS2;                   //JCL
20800: 900 uPSI_Jcl8087 //math coprocessor       //JCL
20801: 901 uPSI_JvAddPrinter                     //JCL
20802: 902 uPSI_JvCabFile                        //JCL
20803: 903 uPSI_JvDataEmbedded;                  //JCL
20804: 904 unit uPSI_U_HexView;                  //mX4
20805: 905 uPSI_UWavein4,                        //mX4
20806: 906 uPSI_AMixer,                          //mX4
20807: 907 uPSI_JvaScrollText,                   //mX4
20808: 908 uPSI_JvArrow,                         //mX4
20809: 909 unit uPSI_UrlMon;                     //mX4
20810: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas'  //mX4
20811: 911 unit uPSI_U_Oscilloscope4; //TOscfrmMain;  //DFF
20812: 912 unit uPSI_DFFUtils;                    //DFF
20813: 913 unit uPSI_MathsLib;                    //DFF
20814: 914 uPSI_UIntList;                        //DFF
20815: 915 uPSI_UGetParens;                      //DFF
20816: 916 unit uPSI_UGeometry;                  //DFF
20817: 917 unit uPSI_UAstronomy;                 //DFF
20818: 918 unit uPSI_UCardComponentV2;           //DFF
20819: 919 unit uPSI_UTGraphSearch;              //DFF
20820: 920 unit uPSI_UParser10;                  //DFF
20821:
20822:
20823:
20824:
20825:
20826: ////////////////////////////////////////////////////////////////////////////
20827: //Form Template Library FTL
20828: ////////////////////////////////////////////////////////////////////////////
20829:
20830: 30 FTL For Form Building out of the Script, eg. 399_form_templates.txt
20831:
20832: 045 unit uPSI_VListView                     TFormListView;
20833: 263 unit uPSI_JvProfiler32;                 TProfReport
20834: 270 unit uPSI_ImgList;                      TCustomImageList
20835: 278 unit uPSI_JvImageWindow;                TJvImageWindow
20836: 317 unit uPSI_JvParserForm;                 TJvHTMLParserForm
20837: 497 unit uPSI_DebugBox;                     TDebugBox
20838: 513 unit uPSI_ImageWin;                     TImageForm, TImageForm2
20839: 514 unit uPSI_CustomDrawTreeView;           TCustomDrawForm
20840: 515 unit uPSI_GraphWin;                     TGraphWinForm
20841: 516 unit uPSI_actionMain;                   TActionForm
20842: 518 unit uPSI_CtlPanel;                     TAppletApplication
20843: 529 unit uPSI_MDIEdit;                      TEditForm
```

```
20844: 535 unit uPSI_CoolMain; {browser}              TWebMainForm
20845: 538 unit uPSI_frmExportMain;                   TSynexportForm
20846: 585 unit uPSI_usniffer; {//PortScanForm}       TSniffForm
20847: 600 unit uPSI_ThreadForm;                      TThreadSortForm;
20848: 618 unit uPSI_delphi_arduino_Unit1;            TLEDForm
20849: 620 unit uPSI_fplotMain;                       TfplotForm1
20850: 660 unit uPSI_JvDBQueryParamsForm;             TJvQueryParamsDialog
20851: 677 unit uPSI_OptionsFrm;                      TfrmOptions;
20852: 718 unit uPSI_MonForm;                         TMonitorForm
20853: 742 unit uPSI_SimplePortMain;                  TPortForm1
20854: 770 unit uPSI_ovccalc;                        TOvcCalculator  //widget
20855: 810 unit uPSI_DbExcept;                       TDbEngineErrorDlg
20856: 812 unit uPSI_GL_actorUnit1;                  TglActorForm1  //OpenGL Robot
20857: 846 unit uPSI_synhttp_daemon;                 TTCPHttpDaemon, TTCPHttpThrd, TPingThread
20858: 867 unit uPSI_Debug;                          TfrmDebug
20859: 904 unit uPSI_U_HexView;                      THexForm2
20860: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)   TOscfrmMain
20861:
20862:
20863:
20864: ex.:with TEditForm.create(self) do begin
20865:       caption:= 'Template Form Tester';
20866:       FormStyle:= fsStayOnTop;
20867:       with editor do begin
20868:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
20869:         SelStart:= 0;
20870:         Modified:= False;
20871:       end;
20872:     end;
20873:   with TWebMainForm.create(self) do begin
20874:     URLs.Text:= 'http://www.kleiner.ch';
20875:     URLsClick(self); Show;
20876:   end;
20877:   with TSynexportForm.create(self) do begin
20878:     Caption:= 'Synexport HTML RTF tester';
20879:     Show;
20880:   end;
20881:   with TThreadSortForm.create(self) do begin
20882:      showmodal; free;
20883:   end;
20884:
20885:   with TCustomDrawForm.create(self) do begin
20886:        width:=820; height:=820;
20887:        image1.height:= 600; //add properties
20888:        image1.picture.bitmap:= image2.picture.bitmap;
20889:        //SelectionBackground1Click(self) CustomDraw1Click(self);
20890:        Background1.click;
20891:        bitmap1.click;
20892:        Tile1.click;
20893:        Showmodal;
20894:        Free;
20895:      end;
20896:
20897:    with TfplotForm1.Create(self) do begin
20898:      BtnPlotClick(self);
20899:      Showmodal; Free;
20900:    end;
20901:
20902:  with TOvcCalculator.create(self) do begin
20903:     parent:= aForm;
20904:   //free;
20905:     setbounds(550,510,200,150);
20906:     displaystr:= 'maXcalc';
20907:   end;
20908:
20909:    with THexForm2.Create(self) do begin
20910:      ShowModal;
20911:      Free;
20912:    end;
20913:
20914:
20915:
20916:
20917: ////////////////////////////////////////////////////////////////////////////
20918: All maXbox Tutorials Table of Content 2014
20919: ////////////////////////////////////////////////////////////////////////////
20920:     Tutorial 00 Function-Coding  (Blix the Programmer)
20921:     Tutorial 01 Procedural-Coding
20922:     Tutorial 02 OO-Programming
20923:     Tutorial 03 Modular Coding
20924:     Tutorial 04 UML Use Case Coding
20925:     Tutorial 05 Internet Coding
20926:     Tutorial 06 Network Coding
20927:     Tutorial 07 Game Graphics Coding
20928:     Tutorial 08 Operating System Coding
20929:     Tutorial 09 Database Coding
20930:     Tutorial 10 Statistic Coding
20931:     Tutorial 11 Forms Coding
20932:     Tutorial 12 SQL DB Coding
```

```
20933:     Tutorial 13 Crypto Coding
20934:     Tutorial 14 Parallel Coding
20935:     Tutorial 15 Serial RS232 Coding
20936:     Tutorial 16 Event Driven Coding
20937:     Tutorial 17 Web Server Coding
20938:     Tutorial 18 Arduino System Coding
20939:     Tutorial 18_3 RGB LED System Coding
20940:     Tutorial 19 WinCOM /Arduino Coding
20941:     Tutorial 20 Regular Expressions RegEx
20942:     Tutorial 21 Android Coding (coming 2013)
20943:     Tutorial 22 Services Programming
20944:     Tutorial 23 Real Time Systems
20945:     Tutorial 24 Clean Code
20946:     Tutorial 25 maXbox Configuration I+II
20947:     Tutorial 26 Socket Programming with TCP
20948:     Tutorial 27 XML & TreeView
20949:     Tutorial 28 DLL Coding (available)
20950:     Tutorial 29 UML Scripting (2014)
20951:     Tutorial 30 Web of Things (coming 2014)
20952:     Tutorial 31 Closures (coming 2014)
20953:     Tutorial 32 SQL Firebird (coming 2014)
20954:
20955:
20956: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
20957: using Docu for this file is maxbox_functions_all.pdf
20958: PEP - Pascal Education Program  Lib Lab
20959:
20960: http://stackoverflow.com/tags/pascalscript/hot
20961: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
20962: http://sourceforge.net/projects/maXbox   #locs:51620
20963: http://sourceforge.net/apps/mediawiki/maXbox
20964: http://www.blaisepascal.eu/
20965: https://github.com/maxkleiner/maXbox3.git
20966: http://www.heise.de/download/maxbox-1176464.html
20967: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
20968: https://www.facebook.com/pages/Programming-maXbox/166844836691703
20969:
20970:   ---- bigbitbox code_cleared_checked----
```