```
 1: ********************************************************************
 2:  Constructor Function and Procedure List of maXbox 3.9.9
 3: ********************************************************************
 4:
 5: Help Extraxt of EXE Functions of maxbox3.exe  inc. API HEX in the BOX
 6: --------------------------------------------------------------------
 7:
 8: File Size of EXE: 12787712 V3.9.9.6 August 2013 To 3.9.9.6
 9: *************Now the Funclist*****************************************
10: Funclist Function :  6862 6450  (/5987 /5681)
11: *************Now the Proclist*****************************************
12: Proclist Procedure Size is: 4059  (3293 /3189 /2985)
13: *************Now Constructors*****************************************
14: Constrlist Constructor Size is: 661 /513 /494 (469)
15:  head:max: APSN21: 06.08.2013 19:51:55
16:  E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
17: --------------------------------------------------------------------
18: Funclist total Size all is: 11546! (9952) Constructor, Function and Procedure
19: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 12190
20: ASize of EXE: 12787712 (11504640) (11399680) (11129344) (10644480)
21: SHA1 Hash of maXbox 3.9.9.6: 76ECD732E1531108B35457D56282B0BBBBB20BAA
22: --------------------------------------------------------------------
23: --------------------------------------------------------------------
24:
25:
26: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
27: Function *************Now the Funclist*****************
28: function  GetResStringChecked(Ident: string; const Args: array of const): string
29: Function ( Index : Longint) : Integer
30: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
31: Function _CheckAutoResult( ResultCode : HResult) : HResult
32: function _T(Name: tbtString): Variant;
33: function ABNFToText(const AText : String) : String
34: Function Abs(e : Extended) : Extended;
35: Function Ackermann( const A, B : Integer) : Integer
36: Function AcquireLayoutLock : Boolean
37: Function ActionByName( const AName : string) : TWebActionItem
38: Function ACTIVEBUFFER : PCHAR
39: Function Add : TAggregate
40: function Add : TCollectionItem
41: Function Add : TColumn
42: Function Add : TComboExItem
43: Function Add : TCookie
44: Function Add : TCoolBand
45: Function Add : TFavoriteLinkItem
46: Function Add : TFileTypeItem
47: Function Add : THeaderSection
48: Function Add : THTMLTableColumn
49: Function Add : TIdEMailAddressItem
50: Function Add : TIdMessagePart
51: Function Add : TIdUserAccount
52: Function Add : TListColumn
53: Function Add : TListItem
54: Function Add : TStatusPanel
55: Function Add : TTaskDialogBaseButtonItem
56: Function Add : TWebActionItem
57: Function Add : TWorkArea
58: Function Add( AClass : TClass) : Integer
59: Function Add( AComponent : TComponent) : Integer
60: Function Add( AItem, AData : Integer) : Integer
61: Function Add( AItem, AData : Pointer) : Pointer
62: Function Add( AItem, AData : TObject) : TObject
63: Function Add( AObject : TObject) : Integer
64: Function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
65: Function Add( const S : WideString) : Integer
66: Function Add( Image, Mask : TBitmap) : Integer
67: Function Add( Index : LongInt; const Text : string) : LongInt
68: Function Add( Sibling : TTreeNode; const S : string) : TTreeNode
69: Function Add(const S: string): Integer
70: function Add(S: string): Integer;
71: Function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
72: Function ADDCHILD : TFIELDDEF
73: Function AddChild( Index : LongInt; const Text : string) : LongInt
74: Function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
75: Function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
76: Function AddChildObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
77: Function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
78: Function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
79: Function ADDFIELDDEF : TFIELDDEF
80: Function AddFileExtIfNecessary( AFileName, AExt : string) : string
81: Function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
82: Function AddIcon( Image : TIcon) : Integer
83: Function AddImage( Value : TCustomImageList; Index : Integer) : Integer
84: Function ADDINDEXDEF : TINDEXDEF
85: Function AddItem( const Caption: String; const ImageIndex,SelectedImageIndex, OverlayImageIndex,Indent:
    Integer; Data : Pointer) : TComboExItem
```

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

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

```
255: Function BitsLowest5( X : Integer) : Integer;
256: Function BitsLowest6( X : Int64) : Integer;
257: Function BitsNeeded( const X : Byte) : Integer;
258: Function BitsNeeded1( const X : Word) : Integer;
259: Function BitsNeeded2( const X : Integer) : Integer;
260: Function BitsNeeded3( const X : Int64) : Integer;
261: Function BlueComponent( const Color32 : TColor32) : Integer
262: Function BooleanToInteger( const Pb : Boolean) : Integer
263: Function BoolToStr(B: Boolean; UseBoolStrs: Boolean): string)
264: Function BoolToStr1(value : boolean) : string;
265: Function booltoint( aBool : Boolean) : LongInt');
266: Function inttobool( aInt : LongInt) : Boolean');
267: Function Bounds( ALeft, ATop, AWidth, AHeight : Integer) : TRect
268: function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect)
269: Function BreakApart( BaseString, BreakString : string; StringList : TStrings) : TStrings
270: Function BrightColor( const Color : TColor; const Pct : Single) : TColor
271: Function BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte
272: Function BufferRequest( Length : Integer) : TStream
273: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
274: Function Buttons : PTaskDialogButton
275: Function BytesPerScanline( PixelsPerScanline, BitsPerPixel, Alignment : Longint) : Longint
276: Function BytesToCardinal( const AValue : TIdBytes; const AIndex : Integer) : Cardinal
277: Function BytesToChar( const AValue : TIdBytes; const AIndex : Integer) : Char
278: Function BytesToInt64( const AValue : TIdBytes; const AIndex : Integer) : Int64
279: Function BytesToInteger( const AValue : TIdBytes; const AIndex : Integer) : Integer
280: Function BytesToIPv6( const AValue : TIdBytes; const AIndex : Integer) : TIdIPv6Address
281: Function BytesToShort( const AValue : TIdBytes; const AIndex : Integer) : Short
282: Function BytesToString(ABytes:TIdBytes; AStartIndex:Integer; AMaxCount:Integer): string;
283: Function BytesToStr(const Value: TBytes): String;
284: Function BytesToWord( const AValue : TIdBytes; const AIndex : Integer) : Word
285: Function ByteToBin(Int: Byte): String;
286: Function ByteToCharIndex( S : string; Index : Integer) : Integer
287: function ByteToCharIndex(const S: string; Index: Integer): Integer)
288: Function ByteToCharLen( S : string; MaxLen : Integer) : Integer
289: Function ByteToCharLen(const S: string; MaxLen: Integer): Integer)
290: Function ByteToHex( const AByte : Byte) : string
291: Function ByteToOctal( const AByte : Byte) : string
292: Function ByteType( S : string; Index : Integer) : TMbcsByteType
293: function ByteType(const S: string; Index: Integer): TMbcsByteType)
294: Function CalcTitleRect( Col : TColumn; ARow : Integer; var MasterCol : TColumn) : TRect
295: Function CalculateDFAFingerprint( oStates : TList) : integer
296: function CallTerminateProcs: Boolean)
297: function CANFOCUS:BOOLEAN
298: Function CanLoad( const Ext : string) : Boolean
299: Function CanParse( AWebRequest : TWebRequest) : Boolean
300: Function CanSave( const Ext : string) : Boolean
301: Function CanStart( cChar : char) : boolean
302: Function CaptureScreen : TBitmap;
303: Function CaptureScreen1( Rec : TRect) : TBitmap;
304: Function CardinalToFourChar( ACardinal : LongWord) : string
305: Function CastSoapToNative(Info:PTypeInfo;const SoapData:WideString;NatData:Pointer;IsNull:Boolean): Boolean
306: Function CastSoapToVariant1( SoapInfo : PTypeInfo; const SoapData : WideString) : Variant;
307: Function Ceil( const X : Extended) : Integer
308: Function Ceil16( X : Integer) : Integer
309: Function Ceil4( X : Integer) : Integer
310: Function Ceil8( X : Integer) : Integer
311: Function Ceiling( const X : Extended) : Integer
312: Function CellRect( ACol, ARow : Longint) : TRect
313: Function CelsiusToFahrenheit( const AValue : Double) : Double
314: Function CenterPoint( const Rect : TRect) : TPoint
315: function CenterPoint(const Rect: TRect): TPoint)
316: Function ChangeFileExt( FileName, Extension : string) : string
317: function ChangeFileExt(const FileName: string; const Extension: string): string)
318: Function CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) : Boolean');
319: Function CharInSet( const Ch : Char; const testSet: TSysCharSet): Boolean');
320: Function CharIsInEOF( const AString : string; ACharPos : Integer) : Boolean
321: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : String) : Boolean
322: Function CharLength( S : String; Index : Integer) : Integer
323: Function CharRange( const AMin, AMax : Char) : String
324: function CharsetToIdent(Charset: Longint; var Ident: string): Boolean)
325: Function CharToBin(vChr: Char): String;');
326: Function CharNext(lpsz: PChar): PChar; stdcall;');
327: Function CharToByteIndex( S : string; Index : Integer) : Integer
328: function CharToByteIndex(const S: string; Index: Integer): Integer)
329: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
330: function CharToByteLen(const S: string; MaxLen: Integer): Integer)
331: Function CharToHex(const APrefix : String; const cc : Char) : shortstring;
332: function CharToHexStr(Value: char): string);
333: function CharToOem(ins, outs: PChar):boolean;
334: function CharToUniCode(Value: Char): string;
335: Function CheckMenuDropdown : Boolean
336: Function CheckMessages : longint
337: Function CheckOpen( Status : DBIResult) : Boolean
338: Function CheckPassword( const APassword : String) : Boolean
339: Function CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt
340: function CheckSynchronize(Timeout: Integer): Boolean
```

```
341: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
342: function ChrA(const a: byte): char;
343: Function ClassIDToProgID(const ClassID: TGUID): string;
344: Function ClassNameIs(const Name: string): Boolean
345: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
346: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
347: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
348: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;
349: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;
350: Function ClearBit5( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
351: Function ClearBit6( const Value : Int64; const Bit : TBitRange) : Int64;
352: function CLIENTTOSCREEN(POINT:TPOINT):TPOINT
353: Function Clipboard : TClipboard
354: Function ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
355: Function ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
356: Function ClipLine( var X1, Y1, X2, Y2 : Integer; const ClipRect : TRect) : Boolean;
357: Function ClipLineToRect( var P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean
358: Function Clone( out stm : IStream) : HResult
359: Function CloneConnection : TSQLConnection
360: Function CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
361: function CLOSEQUERY:BOOLEAN
362: Function CloseVolume( var Volume : THandle) : Boolean
363: Function CloseHandle(Handle: Integer): Integer; stdcall;
364: Function CmdLine: PChar;
365: function CmdShow: Integer;
366: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
367: Function Color32( WinColor : TColor) : TColor32;
368: Function Color321( const Index : Byte; const Palette : TPalette32) : TColor32;
369: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale :BOOLean) : TColor
370: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
371: Function ColorToHTML( const Color : TColor) : String
372: function ColorToIdent(Color: Longint; var Ident: string): Boolean)
373: Function ColorToRGB(color: TColor): Longint
374: function ColorToString(Color: TColor): string)
375: Function ColorToWebColorName( Color : TColor) : string
376: Function ColorToWebColorStr( Color : TColor) : string
377: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn
378: Function Combination(npr, ncr: integer): extended;
379: Function CombinationInt(npr, ncr: integer): Int64;
380: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo');
381: Function CommaAdd( const AStr1, AStr2 : String) : string
382: Function CommercialRound( const X : Extended) : Int64
383: Function Commit( grfCommitFlags : Longint) : HResult
384: Function Compare( const NameExt : string) : Boolean
385: function CompareDate(const A, B: TDateTime): TValueRelationship;
386: Function CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Integer
387: Function CompareFiles(const FN1,FN2 :string; Breathe:TNotifyEvent;BreathingSender:TObject): boolean
388: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean
389: Function CompareStr( S1, S2 : string) : Integer
390: function CompareStr(const S1: string; const S2: string): Integer)
391: function CompareString(const S1: string; const S2: string): Integer)
392: Function CompareText( S1, S2 : string) : Integer
393: function CompareText(const S1: string; const S2: string): Integer)
394: Function CompareTextLike(cWildStr,cStr:string;const cWildChar:char;lCaseSensitive:boolean): boolean
395: function CompareTime(const A, B: TDateTime): TValueRelationship;
396: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
397: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
398: Function ComponentTypeToString( const ComponentType : DWORD) : string
399: Function CompToCurrency( Value : Comp) : Currency
400: Function CompToDouble( Value : Comp) : Double
401: function ComputeFileCRC32(const FileName : String) : Integer;
402: function ComputeSHA256(astr: string; amode: char): string)');  //mode F:File, S:String
403: function ComputeSHA512(astr: string; amode: char): string)');  //mode F:File, S:String
404: Function Concat(s: string): string
405: Function ConnectAndGetAll : string
406: Function Connected : Boolean
407: function constrain(x, a, b: integer): integer;
408: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
409: Function ConstraintsDisabled : Boolean
410: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
411: Function ContainsState( oState : TniRegularExpressionState) : boolean
412: Function ContainsStr( const AText, ASubText : string) : Boolean
413: Function ContainsText( const AText, ASubText : string) : Boolean
414: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean
415: Function Content : string
416: Function ContentFromStream( Stream : TStream) : string
417: Function ContentFromString( const S : string) : string
418: Function CONTROLSDISABLED : BOOLEAN
419: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
420: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
421: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
422: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
423: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
424: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
425: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
426: Function ConvTypeToDescription( const AType : TConvType) : string
```

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

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

```
587: Function DefaultCaption : string
588: Function DefaultColor : TColor
589: Function DefaultFont : TFont
590: Function DefaultImeMode : TImeMode
591: Function DefaultImeName : TImeName
592: Function DefaultReadOnly : Boolean
593: Function DefaultWidth : Integer
594: Function DegMinSecToFloat( const Degs, Mins, Secs : Float) : Float
595: Function DegToCycle( const Degrees : Extended) : Extended
596: Function DegToGrad( const Degrees : Extended) : Extended
597: Function DegToGrad( const Value : Extended) : Extended;
598: Function DegToGrad1( const Value : Double) : Double;
599: Function DegToGrad2( const Value : Single) : Single;
600: Function DegToRad( const Degrees : Extended) : Extended
601: Function DegToRad( const Value : Extended) : Extended;
602: Function DegToRad1( const Value : Double) : Double;
603: Function DegToRad2( const Value : Single) : Single;
604: Function DelChar( const pStr : string; const pChar : Char) : string
605: Function DelEnvironmentVar( const Name : string) : Boolean
606: Function Delete( const MsgNum : Integer) : Boolean
607: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
608: Function DeleteFile(const FileName: string): boolean)
609: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS) : Boolean
610: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
611: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
612: Function DelSpace( const pStr : string) : string
613: Function DelString( const pStr, pDelStr : string) : string
614: Function DelTree( const Path : string) : Boolean
615: Function Depth : Integer
616: Function Description : string
617: Function DescriptionsAvailable : Boolean
618: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
619: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
620: Function DescriptionToConvType1(const AFamily:TConvFamily;const ADescription:string;out
     AType:TConvType):Boolean;
621: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
622: Function DialogsToPixelsX( const Dialogs : Word) : Word
623: Function DialogsToPixelsY( const Dialogs : Word) : Word
624: Function Digits( const X : Cardinal) : Integer
625: Function DirectoryExists( const Name : string) : Boolean
626: Function DirectoryExists( Directory : string) : Boolean
627: Function DiskFree( Drive : Byte) : Int64
628: function DiskFree(Drive: Byte): Int64)
629: Function DiskInDrive( Drive : Char) : Boolean
630: Function DiskSize( Drive : Byte) : Int64
631: function DiskSize(Drive: Byte): Int64)
632: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
633: Function DispatchEnabled : Boolean
634: Function DispatchMask : TMask
635: Function DispatchMethodType : TMethodType
636: Function DISPATCHPOPUP( AHANDLE : HMENU) : BOOLEAN
637: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse) : Boolean
638: Function DisplayCase( const S : String) : String
639: Function DisplayRect( Code : TDisplayCode) : TRect
640: Function DisplayRect( TextOnly : Boolean) : TRect
641: Function DisplayStream( Stream : TStream) : string
642: TBufferCoord', 'record Char : integer; Line : integer; end');
643: TDisplayCoord', 'record Column : integer; Row : integer; end');
644: Function DisplayCoord( AColumn, ARow : Integer) : TDisplayCoord');
645: Function BufferCoord( AChar, ALine : Integer) : TBufferCoord');
646: Function DomainName( const AHost : String) : String
647: Function DosPathToUnixPath( const Path : string) : string
648: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer) : Boolean;
649: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
650: Function DoubleToBcd( const AValue : Double) : TBcd;
651: Function DoubleToHex( const D : Double) : string
652: Function DoUpdates : Boolean
653: function Dragging: Boolean;
654: Function DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL');
655: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect) : BOOL');
656: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL');
657: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UINT) : BOOL');
658: {Works like InputQuery but displays 2edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
659: Function DualInputQuery(const ACaption,Prompt1,Prompt2:string;var AValue1,
     AValue2:string;PasswordChar:Char= #0):Boolean;
660: Function DupeString( const AText : string; ACount : Integer) : string
661: Function Edit : Boolean
662: Function EditCaption : Boolean
663: Function EditText : Boolean
664: Function EditFolderList( Folders : TStrings) : Boolean');
665: Function Elapsed( const Update : Boolean) : Cardinal
666: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
667: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
668: Function EncodeDate( Year, Month, Day : Word) : TDateTime
669: function EncodeDate(Year, Month, Day: Word): TDateTime;
670: Function EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
```

```
671: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : TDateTime
672: Function EncodeDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond: Word): TDateTime
673: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
674: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word) : TDateTime
675: Function EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
676: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
677: Function EndIP : String
678: Function EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
679: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
680: Function EndOfAMonth( const AYear, AMonth : Word) : TDateTime
681: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
682: Function EndOfAYear( const AYear : Word) : TDateTime
683: Function EndOfTheDay( const AValue : TDateTime) : TDateTime
684: Function EndOfTheMonth( const AValue : TDateTime) : TDateTime
685: Function EndOfTheWeek( const AValue : TDateTime) : TDateTime
686: Function EndOfTheYear( const AValue : TDateTime) : TDateTime
687: Function EndPeriod( const Period : Cardinal) : Boolean
688: Function EndsStr( const ASubText, AText : string) : Boolean
689: Function EndsText( const ASubText, AText : string) : Boolean
690: Function EnsureMsgIDBrackets( const AMsgID : String) : String
691: Function EnsureRange( const AValue, AMin, AMax : Integer) : Integer;
692: Function EnsureRange1( const AValue, AMin, AMax : Int64) : Int64;
693: Function EnsureRange2( const AValue, AMin, AMax : Double) : Double;
694: Function EOF: boolean
695: Function EOln: boolean
696: Function EqualRect( const R1, R2 : TRect) : Boolean
697: function EqualRect(const R1, R2: TRect): Boolean)
698: Function Equals( Strings : TWideStrings) : Boolean
699: function Equals(Strings: TStrings): Boolean;
700: Function EqualState( oState : TniRegularExpressionState) : boolean
701: Function ErrOutput: Text)
702: function ExceptionParam: String;
703: function ExceptionPos: Cardinal;
704: function ExceptionProc: Cardinal;
705: function ExceptionToString(er: TIFException; Param: String): String;
706: function ExceptionType: TIFException;
707: Function ExcludeTrailingBackslash( S : string) : string
708: function ExcludeTrailingBackslash(const S: string): string)
709: Function ExcludeTrailingPathDelimiter( const APath : string) : string
710: Function ExcludeTrailingPathDelimiter( S : string) : string
711: function ExcludeTrailingPathDelimiter(const S: string): string)
712: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): boolean;
713: Function ExecProc : Integer
714: Function ExecSQL : Integer
715: Function ExecSQL( ExecDirect : Boolean) : Integer
716: Function Execute : _Recordset;
717: Function Execute : Boolean
718: Function Execute : Boolean;
719: Function Execute( const SQL : string; Params : TParams; Cache : Boolean; cursor : phDBICur) : Integer
720: Function Execute( const SQL : WideString; Params : TParams; ResultSet : TPSResult) : Integer
721: Function Execute( ParentWnd : HWND) : Boolean
722: Function Execute1(constCommandText:WideString;const CommandType:TCommandType;const
     ExecuteOptions:TExecuteOptions): _Recordset;
723: Function Execute1( const Parameters : OleVariant) : _Recordset;
724: Function Execute1( ParentWnd : HWND) : Boolean;
725: Function Execute2( var RecordsAffected : Integer; const Parameters : OleVariant) : _Recordset;
726: Function ExecuteAction( Action : TBasicAction) : Boolean
727: Function ExecuteDirect( const SQL : WideString) : Integer
728: Function ExecuteFile(const FileName:string;const Params:string;const DefaultDir:string;ShowCmd:Integer):
     THandle
729: Procedure ExecuteThread2(afunc:TThreadFunction2;thrOK:boolean);AddTypeS('TThreadFunction2','procedure');
730: Function CreateThread2(ThreadFunc: TThreadFunction2) : THandle');
731: function ExeFileIsRunning(ExeFile: string): boolean;
732: function ExePath: string;
733: function ExePathName: string;
734: Function Exists( AItem : Pointer) : Boolean
735: Function ExitWindows( ExitCode : Cardinal) : Boolean
736: function Exp(x: Extended): Extended;
737: Function ExpandEnvironmentVar( var Value : string) : Boolean
738: Function ExpandFileName( FileName : string) : string
739: function ExpandFileName(const FileName: string): string)
740: Function ExpandUNCFileName( FileName : string) : string
741: function ExpandUNCFileName(const FileName: string): string)
742: Function ExpJ( const X : Float) : Float;
743: Function Exsecans( X : Float) : Float
744: Function Extract( const AByteCount : Integer) : string
745: Function Extract( Item : TClass) : TClass
746: Function Extract( Item : TComponent) : TComponent
747: Function Extract( Item : TObject) : TObject
748: Function ExtractFileDir( FileName : string) : string
749: function ExtractFileDir(const FileName: string): string)
750: Function ExtractFileDrive( FileName : string) : string
751: function ExtractFileDrive(const FileName: string): string)
752: Function ExtractFileExt( FileName : string) : string
753: function ExtractFileExt(const FileName: string): string)
754: Function ExtractFileExtNoDot( const FileName : string) : string
```

```
755: Function ExtractFileExtNoDotUpper( const FileName : string) : string
756: Function ExtractFileName( FileName : string) : string
757: function ExtractFileName(const filename: string):string;
758: Function ExtractFilePath( FileName : string) : string
759: function ExtractFilePath(const filename: string):string;
760: Function ExtractRelativePath( BaseName, DestName : string) : string
761: function ExtractRelativePath(const BaseName: string; const DestName: string): string)
762: Function ExtractShortPathName( FileName : string) : string
763: function ExtractShortPathName(const FileName: string): string)
764: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;Strings: TStrings): Integer
765: function ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings): Integer)
766: Function Fact(numb: integer): Extended;
767: Function FactInt(numb: integer): int64;
768: Function Factorial( const N : Integer) : Extended
769: Function FahrenheitToCelsius( const AValue : Double) : Double
770: function FalseBoolStrs: array of string
771: Function Fetch(var AInput:string;const ADelim:string;const ADelete:Bool;const ACaseSensitive:Bool):string
772: Function FetchCaseInsensitive(var AInput:string; const ADelim:string; const ADelete:Boolean): string
773: Function Fibo(numb: integer): Extended;
774: Function FiboInt(numb: integer): Int64;
775: Function Fibonacci( const N : Integer) : Integer
776: Function FIELDBYNAME( const FIELDNAME : STRING) : TFIELD
777: Function FIELDBYNAME( const FIELDNAME : WIDESTRING) : TFIELD
778: Function FIELDBYNAME( const NAME : String) : TFIELD
779: Function FIELDBYNAME( const NAME : String) : TFIELDDEF
780: Function FIELDBYNUMBER( FIELDNO : INTEGER) : TFIELD
781: Function FileAge( FileName : string) : Integer
782: Function FileAge(const FileName: string): integer)
783: Function FileCompareText( const A, B : String) : Integer
784: Function FileContains(const FileName:string;Text:string;CaseSensitive:Bool;ExceptionOnError:Bool): Boolean
785: Function FileCreate( FileName : string) : Integer;
786: Function FileCreate(const FileName: string): integer)
787: Function FileCreateTemp( var Prefix : string) : THandle
788: Function FileDateToDateTime( FileDate : Integer) : TDateTime
789: function FileDateToDateTime(FileDate: Integer): TDateTime;
790: Function FileExists( const FileName : string) : Boolean
791: Function FileExists( FileName : string) : Boolean
792: function fileExists(const FileName: string): Boolean;
793: Function FileGetAttr( FileName : string) : Integer
794: Function FileGetAttr(const FileName: string): integer)
795: Function FileGetDate( Handle : Integer) : Integer
796: Function FileGetDate(handle: integer): integer
797: Function FileGetDisplayName( const FileName : string) : string
798: Function FileGetSize( const FileName : string) : Integer
799: Function FileGetTempName( const Prefix : string) : string
800: Function FileGetTypeName( const FileName : string) : string
801: Function FileIsReadOnly( FileName : string) : Boolean
802: Function FileLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
803: Function FileOpen( FileName : string; Mode : LongWord) : Integer
804: Function FileOpen(const FileName: string; mode:integer): integer)
805: Function FileRead(handle: integer; Buffer: PChar; count: LongWord): integer
806: Function FileSearch( Name, DirList : string) : string
807: Function FileSearch(const Name, dirList: string): string)
808: Function FileSeek( Handle : Integer; Offset : Int64; Origin : Integer) : Int64;
809: Function FileSeek( Handle, Offset, Origin : Integer) : Integer;
810: Function FileSeek(handle, offset, origin: integer): integer
811: Function FileSetAttr( FileName : string; Attr : Integer) : Integer
812: function FileSetAttr(const FileName: string; Attr: Integer): Integer)
813: Function FileSetDate(FileName : string; Age : Integer) : Integer;
814: Function FileSetDate(handle: integer; age: integer): integer
815: Function FileSetDate2(FileHandle : Integer; Age : Integer) : Integer;
816: Function FileSetDateH( Handle : Integer; Age : Integer) : Integer;
817: Function FileSetReadOnly( FileName : string; ReadOnly : Boolean) : Boolean
818: Function FileSize( const FileName : string) : int64
819: Function FileSizeByName( const AFilename : string) : Longint
820: function FileWrite(Handle: Integer; const Buffer: pChar; Count: LongWord): Integer)
821: Function FilterSpecArray : TComdlgFilterSpecArray
822: Function FIND( ACAPTION : String) : TMENUITEM
823: Function Find( AItem : Pointer; out AData : Pointer) : Boolean
824: Function FIND( const ANAME : String) : TNAMEDITEM
825: Function Find( const DisplayName : string) : TAggregate
826: Function Find( const Item : TBookmarkStr; var Index : Integer) : Boolean
827: Function FIND( const NAME : String) : TFIELD
828: Function FIND( const NAME : String) : TFIELDDEF
829: Function FIND( const NAME : String) : TINDEXDEF
830: Function Find( const S : WideString; var Index : Integer) : Boolean
831: function Find(S:String;var Index:Integer):Boolean
832: Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
833: Function FindBand( AControl : TControl) : TCoolBand
834: Function FindBoundary( AContentType : string) : string
835: Function FindButton( AModalResult : TModalResult) : TTaskDialogBaseButtonItem
836: Function FindCaption(StartIndex: Integer;Value: string; Partial,Inclusive,Wrap: Boolean): TListItem
837: Function FindCdLineSwitch( Switch : string; IgnoreCase : Boolean) : Boolean;
838: Function FindClose2(FindFile: Integer): LongBool; stdcall;
839: Function FindCmdLineSwitch( Switch : string; Chars : TSysCharSet; IgnoreCase : Boolean) : Boolean;
840: Function FindCmmdLineSwitch( Switch : string) : Boolean;
```

```
841: function FindComponent(AName: String): TComponent;
842: function FindComponent(vlabel: string): TComponent;
843: function FindComponent2(vlabel: string): TComponent;
844: function FindControl(Handle: HWnd): TWinControl;
845: Function FindData( StartIndex : Integer; Value : Pointer; Inclusive, Wrap : Boolean) : TListItem
846: Function FindDatabase( const DatabaseName : string) : TDatabase
847: function FindDragTarget(const Pos: TPoint; AllowDisabled: Boolean): TControl;
848: Function FINDFIELD( const FIELDNAME : STRING) : TFIELD
849: Function FINDFIELD( const FIELDNAME : WideString) : TFIELD
850: Function FINDFIRST : BOOLEAN
851: TJvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
852:   sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms, sfRecent, sfSendTo,
     sfStartMenu, stStartUp, sfTemplates);
853:  FFolder: array [TJvSpecialFolder] of Integer =
854:     (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
855:      CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
856:      CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
857:      CSIDL_STARTUP, CSIDL_TEMPLATES);
858: Function FindFilesDlg(StartIn: string; SpecialFolder: TJvSpecialFolder; UseFolder: Boolean): Boolean);
859: function Findfirst(const filepath: string; attr: integer): integer;
860: function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer)
861: Function FindFirstNotOf( AFind, AText : String) : Integer
862: Function FindFirstOf( AFind, AText : String) : Integer
863: Function FindImmediateTransitionOn( cChar : char) : TniRegularExpressionState
864: Function FINDINDEXFORFIELDS( const FIELDS : String) : TINDEXDEF
865: Function FindInstanceOf( AClass : TClass; AExact : Boolean; AStartAt : Integer) : Integer
866: Function FINDITEM( VALUE : INTEGER; KIND : TFINDITEMKIND) : TMENUITEM
867: function FindItemId( Id : Integer) : TCollectionItem
868: Function FindKey( const KeyValues : array of const) : Boolean
869: Function FINDLAST : BOOLEAN
870: Function FindLineControl( ComponentType, ControlType : DWORD) : TJclMixerLineControl
871: Function FindModuleClass( AClass : TComponentClass) : TComponent
872: Function FindModuleName( const AClass : string) : TComponent
873: Function FINDNEXT : BOOLEAN
874: function FindNext: integer;
875: function FindNext2(var F: TSearchRec): Integer)
876: Function FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible : Boolean) : TTabSheet
877: Function FindNextToSelect : TTreeNode
878: Function FINDPARAM( const VALUE : String) : TPARAM
879: Function FindParam( const Value : WideString) : TParameter
880: Function FINDPRIOR : BOOLEAN
881: Function FindResource( ModuleHandle : HMODULE; ResourceName, ResourceType : PChar) : TResourceHandle
882: Function FindSession( const SessionName : string) : TSession
883: function FindStringResource(Ident: Integer): string)
884: Function FindText( const SearchStr:string;StartPos,Length: Integer; Options: TSearchTypes):Integer
885: Function FindUnusedFileName( const FileName, FileExt, Suffix : AnsiString) : AnsiString
886: function FindVCLWindow(const Pos: TPoint): TWinControl;
887: function FindWindow(C1, C2: PChar): Longint;
888: Function FindInPaths(const fileName,paths: String): String;
889: Function Finger : String
890: Function First : TClass
891: Function First : TComponent
892: Function First : TObject
893: Function FirstDelimiter( const delimiters : string; const Str : String) : integer;
894: Function FirstDelimiter1( const delimiters : WideString; const Str : WideString) : integer;
895: Function FirstInstance( const ATitle : string) : Boolean
896: Function FloatPoint( const X, Y : Float) : TFloatPoint;
897: Function FloatPoint1( const P : TPoint) : TFloatPoint;
898: Function FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint) : Boolean
899: Function FloatRect( const ALeft, ATop, ARight, ABottom : Double) : TFloatRect;
900: Function FloatRect1( const Rect : TRect) : TFloatRect;
901: Function FloatsEqual( const X, Y : Float) : Boolean
902: Function FloatToBin(const D: Double): string;     //doubletohex -> hextobin! in buffer
903: Function FloatToCurr( Value : Extended) : Currency
904: Function FloatToDateTime( Value : Extended) : TDateTime
905: Function FloatToStr( Value : Extended) : string;
906: Function FloatToStr(e : Extended) : String;
907: Function FloatToStrF( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer) : string;
908: function FloatToStrF(Value: Extended; Format: TFloatFormat;Precision:Integer; Digits: Integer): string)
909: Function FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings
     : TFormatSettings) : string;');
910: Function FloatToStrFS( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer;
     FormatSettings : TFormatSettings) : string;');
911: function FloatToText(BufferArg: PChar; const Value: Extended; ValueType: TFloatValue;Format: TFloatFormat;
     Precision,Digits: Integer): Integer)
912: Function Floor( const X : Extended) : Integer
913: Function FloorInt( Value : Integer; StepSize : Integer) : Integer
914: Function FloorJ( const X : Extended) : Integer
915: Function Flush( const Count : Cardinal) : Boolean
916: Function Flush(var t: Text): Integer
917: function FmtLoadStr(Ident: Integer; const Args: array of const): string)
918: function FOCUSED:BOOLEAN
919: Function ForceBackslash( const PathName : string) : string
920: Function ForceDirectories( const Dir : string) : Boolean
921: Function ForceDirectories( Dir : string) : Boolean
922: Function ForceDirectories( Name : string) : Boolean
```

```
 923: Function ForceInBox( const Point : TPoint; const Box : TRect) : TPoint
 924: Function ForceInRange( A, Min, Max : Integer) : Integer
 925: Function ForceInRangeR( const A, Min, Max : Double) : Double
 926: Function ForEach( AProc : TBucketProc; AInfo : Pointer) : Boolean;
 927: Function ForEach1( AEvent : TBucketEvent) : Boolean;
 928: Function ForegroundTask: Boolean
 929: function Format(const Format: string; const Args: array of const): string;
 930: Function FormatBcd( const Format : string; Bcd : TBcd) : string
 931: FUNCTION FormatBigInt(s: string): STRING;
 932: function FormatBuf(var Buffer:PChar;BufLen:Cardinal;const Format:string;FmtLen:Cardinal;const Args:array
      of const): Cardinal
 933: Function FormatCount( iCount : integer; const sSingular : string; const sPlural : string) : string
 934: Function FormatCurr( Format : string; Value : Currency) : string;
 935: function FormatCurr(const Format: string; Value: Currency): string)
 936: Function FormatDateTime( Format : string; DateTime : TDateTime) : string;
 937: function FormatDateTime(const fmt: string; D: TDateTime): string;
 938: Function FormatFloat( Format : string; Value : Extended) : string;
 939: function FormatFloat(const Format: string; Value: Extended): string)
 940: Function FormatFloat( Format : string; Value : Extended) : string;');
 941: Function FormatFloat2( Format : string; Value : Extended; FormatSettings : TFormatSettings) : string;');
 942: Function FormatCurr( Format : string; Value : Currency) : string;');
 943: Function FormatCurr2(Format: string; Value : Currency; FormatSettings : TFormatSettings) : string;');
 944: Function Format2(const Format:string;const Args:array of const;const FSettings:TFormatSettings): string
 945: FUNCTION FormatInt(i: integer): STRING;
 946: FUNCTION FormatInt64(i: int64): STRING;
 947: Function FormatMaskText( const EditMask : string; const Value : string) : string
 948: Function FormatValue( AValue : Cardinal) : string
 949: Function FormatVersionString( const HiV, LoV : Word) : string;
 950: Function FormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
 951: function Frac(X: Extended): Extended);
 952: Function FreeResource( ResData : HGLOBAL) : LongBool
 953: Function FromCommon( const AValue : Double) : Double
 954: function FromCommon(const AValue: Double): Double;
 955: Function FTPGMTDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 956: Function FTPLocalDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean) : String
 957: Function FTPMLSToGMTDateTime( const ATimeStamp : String) : TDateTime
 958: Function FTPMLSToLocalDateTime( const ATimeStamp : String) : TDateTime
 959: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
 960: //Function Funclist Size is: 6444 of mX3.9.8.9
 961: Function FutureValue( const Rate : Extended; NPeriods : Integer; const Payment, PresentValue : Extended;
      PaymentTime : TPaymentTime) : Extended
 962: Function Gauss( const x, Spread : Double) : Double
 963: function Gauss(const x,Spread: Double): Double;
 964: Function GCD(x, y : LongInt) : LongInt;
 965: Function GCDJ( X, Y : Cardinal) : Cardinal
 966: Function GDAL: LongWord
 967: Function GdiFlush : BOOL');
 968: Function GdiSetBatchLimit( Limit : DWORD) : DWORD');
 969: Function GdiGetBatchLimit : DWORD');
 970: Function GenerateHeader : TIdHeaderList
 971: Function GeometricMean( const X : TDynFloatArray) : Float
 972: Function Get( AURL : string) : string;
 973: Function Get2( AURL : string) : string;
 974: Function Get8087CW : Word
 975: function GetActiveOleObject(const ClassName: String): IDispatch;
 976: Function GetAliasDriverName( const AliasName : string) : string
 977: Function GetAPMBatteryFlag : TAPMBatteryFlag
 978: Function GetAPMBatteryFullLifeTime : DWORD
 979: Function GetAPMBatteryLifePercent : Integer
 980: Function GetAPMBatteryLifeTime : DWORD
 981: Function GetAPMLineStatus : TAPMLineStatus
 982: Function GetAppdataFolder : string
 983: Function GetAppDispatcher : TComponent
 984: function GetArrayLength: integer;
 985: Function GetASCII: string;
 986: Function GetASCIILine: string;
 987: Function GetAsHandle( Format : Word) : THandle
 988: function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
 989: Function GetBackupFileName( const FileName : string) : string
 990: Function GetBBitmap( Value : TBitmap) : TBitmap
 991: Function GetBIOSCopyright : string
 992: Function GetBIOSDate : TDateTime
 993: Function GetBIOSExtendedInfo : string
 994: Function GetBIOSName : string
 995: Function getBitmap(apath: string): TBitmap;
 996: Function GetBitmap( Index : Integer; Image : TBitmap) : Boolean  //object
 997: Function getBitMapObject(const bitmappath: string): TBitmap;
 998: Function GetButtonState( Button : TPageScrollerButton) : TPageScrollerButtonState
 999: Function GetCapsLockKeyState : Boolean
1000: function GetCaptureControl: TControl;
1001: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char) : TJclCdTrackInfo;
1002: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char) : string;
1003: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char) : string
1004: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char) : string
1005: Function GetClientThread( ClientSocket : TServerClientWinSocket) : TServerClientThread
1006: Function GetClockValue : Int64
```

```
1007: function getCmdLine: PChar;
1008: function getCmdShow: Integer;
1009: function GetCPUSpeed: Double;
1010: Function GetColField( DataCol : Integer) : TField
1011: Function GetColorBlue( const Color : TColor) : Byte
1012: Function GetColorFlag( const Color : TColor) : Byte
1013: Function GetColorGreen( const Color : TColor) : Byte
1014: Function GetColorRed( const Color : TColor) : Byte
1015: Function GetComCtlVersion : Integer
1016: Function GetCommonAppdataFolder : string
1017: Function GetCommonDesktopdirectoryFolder : string
1018: Function GetCommonFavoritesFolder : string
1019: Function GetCommonFilesFolder : string
1020: Function GetCommonProgramsFolder : string
1021: Function GetCommonStartmenuFolder : string
1022: Function GetCommonStartupFolder : string
1023: Function GetComponent( Owner, Parent : TComponent) : TComponent
1024: Function GetConnectionRegistryFile( DesignMode : Boolean) : string
1025: Function GetCookiesFolder : string
1026: Function GetCPUSpeed( var CpuSpeed : TFreqInfo) : Boolean
1027: Function GetCurrent : TFavoriteLinkItem
1028: Function GetCurrent : TListItem
1029: Function GetCurrent : TTaskDialogBaseButtonItem
1030: Function GetCurrent : TToolButton
1031: Function GetCurrent : TTreeNode
1032: Function GetCurrent : WideString
1033: Function GetCurrentDir : string
1034: function GetCurrentDir: string)
1035: Function GetCurrentFolder : string
1036: Function GETCURRENTRECORD( BUFFER : PCHAR) : BOOLEAN
1037: Function GetCurrentProcessId : TIdPID');
1038: Function GetCurrentThreadHandle : THandle
1039: Function GetCurrentThreadID: LongWord; stdcall;');
1040: Function GetCustomHeader( const Name : string) : String
1041: Function GetDataItem( Value : Pointer) : Longint
1042: Function GetDataLinkFiles( FileNames : TWideStrings; Directory : string) : Integer;
1043: Function GetDataLinkFiles1( FileNames : TStrings; Directory : string) : Integer;
1044: Function GETDATASIZE : INTEGER
1045: Function GetDC(hdwnd: HWND): HDC;
1046: Function GetDefaultFileExt( const MIMEType : string) : string
1047: Function GetDefaults : Boolean
1048: Function GetDefaultSchemaName : WideString
1049: Function GetDefaultStreamLoader : IStreamLoader
1050: Function GetDesktopDirectoryFolder : string
1051: Function GetDesktopFolder : string
1052: Function GetDFAState( oStates : TList) : TniRegularExpressionState
1053: Function GetDirectorySize( const Path : string) : Int64
1054: Function GetDisplayWidth : Integer
1055: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer) : Boolean
1056: Function GetDomainName : string
1057: Function GetDriverRegistryFile( DesignMode : Boolean) : string
1058: function GetDriveType(rootpath: pchar): cardinal;
1059: Function GetDriveTypeStr( const Drive : Char) : string
1060: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1061: Function GetEnumerator : TListItemsEnumerator
1062: Function GetEnumerator : TTaskDialogButtonsEnumerator
1063: Function GetEnumerator : TToolBarEnumerator
1064: Function GetEnumerator : TTreeNodesEnumerator
1065: Function GetEnumerator : TWideStringsEnumerator
1066: Function GetEnvVar( const VarName : string) : string');
1067: Function GetEnvironmentVar( const AVariableName : string) : string
1068: Function GetEnvironmentVariable( const VarName : string) : string');
1069: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean) : Boolean
1070: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean) : Boolean
1071: Function getEnvironmentString: string;
1072: Function GetExceptionHandler : TObject
1073: Function GetFavoritesFolder : string
1074: Function GetFieldByName( const Name : string) : string
1075: Function GetFieldInfo( const Origin : Widestring; var FieldInfo : TFieldInfo) : Boolean
1076: Function GetFieldValue( ACol : Integer) : string
1077: Function GetFileAgeCoherence( const FileName : string) : Boolean
1078: Function GetFileCreation( const FileName : string) : TFileTime
1079: Function GetFileCreationTime( const Filename : string) : TDateTime
1080: Function GetFileInformation( const FileName : string) : TSearchRec
1081: Function GetFileLastAccess( const FileName : string) : TFileTime
1082: Function GetFileLastWrite( const FileName : string) : TFileTime
1083: Function GetFileMIMEType( const AFileName : string) : string
1084: Function GetFileSize( const FileName : string) : Int64
1085: Function GetFileVersion( AFileName : string) : Cardinal
1086: Function GetFileVersion( const AFilename : string) : Cardinal
1087: Function GetFileSize2(Handle: Integer; x: Integer): Integer; stdcall;');
1088: Function GetFilterData( Root : PExprNode) : TExprData
1089: Function getFirstChild : LongInt
1090: Function getFirstChild : TTreeNode
1091: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string) : string
1092: Function GetFirstNode : TTreeNode
```

```
1093: Function GetFontsFolder : string
1094: Function GetFormulaValue( const Formula : string) : Extended');
1095: Function GetFreePageFileMemory : Integer
1096: Function GetFreePhysicalMemory : Integer
1097: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind) : Integer;
1098: Function GetFreeSystemResources1 : TFreeSystemResources;
1099: Function GetFreeVirtualMemory : Integer
1100: Function GetFromClipboard : Boolean
1101: Function GetFullURI( const AOptionalFileds : TIdURIOptionalFieldsSet) : String
1102: Function GetGBitmap( Value : TBitmap) : TBitmap
1103: Function GetGMTDateByName( const AFileName : TIdFileName) : TDateTime
1104: Function GetGroupState( Level : Integer) : TGroupPosInds
1105: Function GetHandle : HWND
1106: Function GETHELPCONTEXT( VALUE : INTEGER; BYCOMMAND : BOOLEAN) : THELPCONTEXT
1107: function GetHexArray(ahexdig: THexArray): THexArray;
1108: Function GetHighLightColor( const Color : TColor; Luminance : Integer) : TColor
1109: function GetHINSTANCE: longword;
1110: Function GetHistoryFolder : string
1111: Function GetHitTestInfoAt( X, Y : Integer) : THitTests
1112: function getHMODULE: longword;
1113: Function GetHostByName(const AComputerName: String): String;
1114: Function GetHostName : string
1115: Function GetHotSpot : TPoint
1116: Function GetHueBitmap( Value : TBitmap) : TBitmap
1117: Function GetImageBitmap : HBITMAP
1118: Function GETIMAGELIST : TCUSTOMIMAGELIST
1119: Function GetIncome( const aNetto : Currency) : Currency
1120: Function GetIncome( const aNetto : Extended) : Extended
1121: Function GetIncome( const aNetto : Extended): Extended
1122: Function GetIncome(const aNetto : Extended) : Extended
1123: function GetIncome(const aNetto: Currency): Currency
1124: Function GetIncome2( const aNetto : Currency) : Currency
1125: Function GetIncome2( const aNetto : Currency): Currency
1126: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string) : string
1127: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN) : TINDEXDEF
1128: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet) : TIndexDef
1129: Function GetInstRes( Instance : THandle; ResType : TResType; const Name : string; Width : Integer;
      LoadFlags : TLoadResources; MaskColor : TColor) : Boolean;
1130: Function GetInstRes1( Instance : THandle; ResType : TResType; ResID : DWORD; Width : Integer; LoadFlags :
      TLoadResources; MaskColor : TColor) : Boolean;
1131: Function GetIntelCacheDescription( const D : Byte) : string
1132: Function GetInteractiveUserName : string
1133: Function GetInternetCacheFolder : string
1134: Function GetInternetFormattedFileTimeStamp( const AFilename : String) : String
1135: Function GetIPAddress( const HostName : string) : string
1136: Function GetIP( const HostName : string) : string
1137: Function GetIPHostByName(const AComputerName: String): String;
1138: Function GetItem( X, Y : Integer) : LongInt
1139: Function GetItemAt( X, Y : Integer) : TListItem
1140: Function GetItemHeight(Font: TFont): Integer;
1141: Function GetItemPath( Index : Integer) : string
1142: Function GetKeyFieldNames( List : TStrings) : Integer;
1143: Function GetKeyFieldNames1( List : TWideStrings) : Integer;
1144: Function GetKeyState( const VirtualKey : Cardinal) : Boolean
1145: Function GetLastChild : LongInt
1146: Function GetLastChild : TTreeNode
1147: Function GetLastDelimitedToken( const cDelim : char; const cStr : string) : string
1148: function GetLastError: Integer
1149: Function GetLAT_CONV_FACTOR: double;');  //for WGS84 power(1 - 1 / 298.257223563, 2);
1150: Function GetLoader( Ext : string) : TBitmapLoader
1151: Function GetLoadFilter : string
1152: Function GetLocalComputerName : string
1153: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char) : Char
1154: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string) : string
1155: Function GetLocalUserName : string
1156: Function GetLoginUsername : WideString
1157: function getLongDayNames: string)
1158: Function GetLongHint(const hint: string): string
1159: function getLongMonthNames: string)
1160: Function GetMacAddresses( const Machine : string; const Addresses : TStrings) : Integer
1161: Function GetMainAppWndFromPid( PID : DWORD) : HWND
1162: Function GetMaskBitmap : HBITMAP
1163: Function GetMaxAppAddress : Integer
1164: Function GetMciErrorMessage( const MciErrNo : MCIERROR) : string
1165: Function GetMemoryLoad : Byte
1166: Function GetMIMEDefaultFileExt( const MIMEType : string) : TIdFileName
1167: Function GetMIMETypeFromFile( const AFile : string) : string
1168: Function GetMIMETypeFromFile( const AFile : TIdFileName) : string
1169: Function GetMinAppAddress : Integer
1170: Function GetModule : TComponent
1171: Function GetModuleHandle( ModuleName : PChar) : HMODULE
1172: Function GetModuleName( Module : HMODULE) : string
1173: Function GetModulePath( const Module : HMODULE) : string
1174: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;');
1175: Function GetCommandLine: PChar; stdcall;');
1176: Function GetMonochromeBitmap( Value : TBitmap) : TBitmap
```

```
1177: Function GetMultiN(aval: integer): string;
1178: Function GetName : String
1179: Function GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1180: Function GetNethoodFolder : string
1181: Function GetNext : TTreeNode
1182: Function GetNextChild( Value : LongInt) : LongInt
1183: Function GetNextChild( Value : TTreeNode) : TTreeNode
1184: Function GetNextDelimitedToken( const cDelim : char; var cStr : String) : String
1185: Function GetNextItem( StartItem: TListItem;Direction:TSearchDirection;States:TItemStates) : TListItem
1186: Function GetNextPacket : Integer
1187: Function getNextSibling : TTreeNode
1188: Function GetNextVisible : TTreeNode
1189: Function GetNode( ItemId : HTreeItem) : TTreeNode
1190: Function GetNodeAt( X, Y : Integer) : TTreeNode
1191: Function GetNodeDisplayWidth( Node : TOutlineNode) : Integer
1192: function GetNumberOfProcessors: longint;
1193: Function GetNumLockKeyState : Boolean
1194: Function GetObjectProperty( Instance : TPersistent; const PropName : string) : TObject
1195: Function GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1196: Function GetOptionalParam( const ParamName : string) : OleVariant
1197: Function GetOSName: string;');
1198: Function GetOSVersion: string;
1199: Function GetOSNumber: string;
1200: Function GetPackageModuleHandle( PackageName : PChar) : HMODULE
1201: function GetPageSize: Cardinal;
1202: Function GetParameterFileName : string
1203: Function GetParams( var OwnerData : OleVariant) : OleVariant
1204: Function GETPARENTCOMPONENT : TCOMPONENT
1205: Function GetParentForm(control: TControl): TForm
1206: Function GETPARENTMENU : TMENU
1207: Function GetPassword : Boolean
1208: Function GetPassword : string
1209: Function GetPersonalFolder : string
1210: Function GetPidFromProcessName( const ProcessName : string) : DWORD
1211: Function GetPosition : TPoint
1212: Function GetPrev : TTreeNode
1213: Function GetPrevChild( Value : LongInt) : LongInt
1214: Function GetPrevChild( Value : TTreeNode) : TTreeNode
1215: Function getPrevSibling : TTreeNode
1216: Function GetPrevVisible : TTreeNode
1217: Function GetPrinthoodFolder : string
1218: Function GetPrivilegeDisplayName( const PrivilegeName : string) : string
1219: Function getProcessList: TStrings;
1220: Function GetProcessId : TIdPID
1221: Function GetProcessNameFromPid( PID : DWORD) : string
1222: Function GetProcessNameFromWnd( Wnd : HWND) : string
1223: Function GetProgramFilesFolder : string
1224: Function GetProgramsFolder : string
1225: Function GetProxy : string
1226: Function GetQuoteChar : WideString
1227: Function GetRate : Double
1228: Function GetRBitmap( Value : TBitmap) : TBitmap
1229: Function GetReadableName( const AName : string) : string
1230: Function GetRecentDocs : TStringList
1231: Function GetRecentFolder : string
1232: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer) : OleVariant;
1233: Function GetRecords1( Count : Integer; out RecsOut : Integer; Options : Integer; const CommandText :
      WideString; var Params, OwnerData : OleVariant) : OleVariant;
1234: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString) : _Recordset
1235: Function GetRegisteredCompany : string
1236: Function GetRegisteredOwner : string
1237: Function GetResource(ResType:TResType;const
      Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor:Boolean
1238: Function GetResourceName( ObjStream : TStream; var AName : string) : Boolean');
1239: Function GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1240: Function GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1241: Function GetRValue( rgb : DWORD) : Byte');
1242: Function GetGValue( rgb : DWORD) : Byte');
1243: Function GetBValue( rgb : DWORD) : Byte');
1244: Function GetCValue( cmyk : COLORREF) : Byte');
1245: Function GetMValue( cmyk : COLORREF) : Byte');
1246: Function GetYValue( cmyk : COLORREF) : Byte');
1247: Function GetKValue( cmyk : COLORREF) : Byte');
1248: Function CMYK( c, m, y, k : Byte) : COLORREF');
1249: Function GetOSName: string;');
1250: Function GetSafeCallExceptionMsg : String
1251: Function GetSaturationBitmap( Value : TBitmap) : TBitmap
1252: Function GetSaveFilter : string
1253: Function GetSaver( Ext : string) : TBitmapLoader
1254: Function GetScrollLockKeyState : Boolean
1255: Function GetSearchString : string
1256: Function GetSelections( AList : TList) : TTreeNode
1257: function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1258: Function GetSendToFolder : string
1259: Function GetServer : IAppServer
1260: Function GetServerList : OleVariant
```

```
1261: Function GetShadowColor( const Color : TColor; Luminance : Integer) : TColor
1262: Function GetShellProcessHandle : THandle
1263: Function GetShellProcessName : string
1264: Function GetShellVersion : Cardinal
1265: function getShortDayNames: string)
1266: Function GetShortHint(const hint: string): string
1267: function getShortMonthNames: string)
1268: Function GetSizeOfFile( const FileName : string) : Int64;
1269: Function GetSizeOfFile1( Handle : THandle) : Int64;
1270: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1271: Function GetStartmenuFolder : string
1272: Function GetStartupFolder : string
1273: Function GetStringProperty( Instance : TPersistent; const PropName : string) : WideString
1274: Function GetSuccessor( cChar : char) : TniRegularExpressionState
1275: Function GetSwapFileSize : Integer
1276: Function GetSwapFileUsage : Integer
1277: Function GetSystemLocale : TIdCharSet
1278: Function GetSystemMetrics( nIndex : Integer) : Integer');
1279: Function GetSystemPathSH(Folder: Integer): TFilename ;');
1280: Function GetTableNameFromQuery( const SQL : Widestring) : Widestring
1281: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1282: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1283: Function GetTasksList( const List : TStrings) : Boolean
1284: Function GetTemplatesFolder : string
1285: Function GetText : PwideChar
1286: function GetText:PChar
1287: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1288: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1289: Function GetTextItem( const Value : string) : Longint
1290: function GETTEXTLEN:INTEGER
1291: Function GetTickCount : Cardinal
1292: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal) : Cardinal
1293: Function GetTicketNr : longint
1294: Function GetTime : Cardinal
1295: Function GetTime : TDateTime
1296: Function GetTimeout : Integer
1297: Function GetTimeStr: String
1298: Function GetTimeString: String
1299: Function getTokenCounts : integer
1300: Function GetTotalPageFileMemory : Integer
1301: Function GetTotalPhysicalMemory : Integer
1302: Function GetTotalVirtualMemory : Integer
1303: Function GetUniqueFileName( const APath, APrefix, AExt : String) : String
1304: Function GetUseNowForDate : Boolean
1305: Function GetUserDomainName( const CurUser : string) : string
1306: Function GetUserName : string
1307: Function GetUserName: string;
1308: Function GetUserObjectName( hUserObject : THandle) : string
1309: Function GetValueBitmap( Value : TBitmap) : TBitmap
1310: Function GetValueMSec : Cardinal
1311: Function GetValueStr : String
1312: Function GetVersionString(FileName: string): string;
1313: Function GetVisibleNode( Index : LongInt) : TOutlineNode
1314: Function GetVolumeFileSystem( const Drive : string) : string
1315: Function GetVolumeName( const Drive : string) : string
1316: Function GetVolumeSerialNumber( const Drive : string) : string
1317: Function GetWebAppServices : IWebAppServices
1318: Function GetWebRequestHandler : IWebRequestHandler
1319: Function GetWindowCaption( Wnd : HWND) : string
1320: Function GetWindowDC(hdwnd: HWND): HDC;
1321: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1322: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1323: Function GetWindowsComputerID : string
1324: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1325: Function GetWindowsFolder : string
1326: Function GetWindowsServicePackVersion : Integer
1327: Function GetWindowsServicePackVersionString : string
1328: Function GetWindowsSystemFolder : string
1329: Function GetWindowsTempFolder : string
1330: Function GetWindowsUserID : string
1331: Function GetWindowsVersion : TWindowsVersion
1332: Function GetWindowsVersionString : string
1333: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1334: Function GMTToLocalDateTime( S : string) : TDateTime
1335: Function GotoKey : Boolean
1336: Function GradToCycle( const Grads : Extended) : Extended
1337: Function GradToDeg( const Grads : Extended) : Extended
1338: Function GradToDeg( const Value : Extended) : Extended;
1339: Function GradToDeg1( const Value : Double) : Double;
1340: Function GradToDeg2( const Value : Single) : Single;
1341: Function GradToRad( const Grads : Extended) : Extended
1342: Function GradToRad( const Value : Extended) : Extended;
1343: Function GradToRad1( const Value : Double) : Double;
1344: Function GradToRad2( const Value : Single) : Single;
1345: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1346: Function GreenComponent( const Color32 : TColor32) : Integer
```

```
1347: function GUIDToString(const GUID: TGUID): string)
1348: Function HandleAllocated : Boolean
1349: function HandleAllocated: Boolean;
1350: Function HandleRequest : Boolean
1351: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1352: Function HarmonicMean( const X : TDynFloatArray) : Float
1353: Function HasAsParent( Value : TTreeNode) : Boolean
1354: Function HASCHILDDEFS : BOOLEAN
1355: Function HasCurValues : Boolean
1356: Function HasExtendCharacter( const s : UTF8String) : Boolean
1357: Function HasFormat( Format : Word) : Boolean
1358: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1359: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1360: Function HashValue(AStream: TStream): LongWord
1361: Function HashValue(AStream: TStream): Word
1362: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1363: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1364: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1365: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1366: Function HashValue16( const ASrc : string) : Word;
1367: Function HashValue16stream( AStream : TStream) : Word;
1368: Function HashValue32( const ASrc : string) : LongWord;
1369: Function HashValue32Stream( AStream : TStream) : LongWord;
1370: Function HasMergeConflicts : Boolean
1371: Function hasMoreTokens : boolean
1372: Function HASPARENT : BOOLEAN
1373: function HasParent: Boolean
1374: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1375: Function HasUTF8BOM( S : TStream) : boolean;
1376: Function HasUTF8BOM1( S : AnsiString) : boolean;
1377: Function Haversine( X : Float) : Float
1378: Function Head( s : string; const subs : string; var tail : string) : string
1379: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1380: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1381: function HELPJUMP(JUMPID:STRING):BOOLEAN
1382: Function HeronianMean( const a, b : Float) : Float
1383: function HexStrToStr(Value: string): string;
1384: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1385: function HexToBin2(HexNum: string): string;
1386: Function HexToDouble( const Hex : string) : Double
1387: function HexToInt(hexnum: string): LongInt;
1388: function HexToStr(Value: string): string;
1389: Function HexifyBlock( var Buffer, BufferSize : Integer) : string');
1390: function Hi(vdat: word): byte;
1391: function HiByte(W: Word): Byte)
1392: function High: Int64;
1393: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1394: function HINSTANCE: longword;
1395: function HiWord(l: DWORD): Word)
1396: function HMODULE: longword;
1397: Function HourOf( const AValue : TDateTime) : Word
1398: Function HourOfTheDay( const AValue : TDateTime) : Word
1399: Function HourOfTheMonth( const AValue : TDateTime) : Word
1400: Function HourOfTheWeek( const AValue : TDateTime) : Word
1401: Function HourOfTheYear( const AValue : TDateTime) : Word
1402: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1403: Function HourSpan( const ANow, AThen : TDateTime) : Double
1404: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1405: Function HTMLDecode( const AStr : String) : String
1406: Function HTMLEncode( const AStr : String) : String
1407: Function HTMLEscape( const Str : string) : string
1408: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1409: Function HTTPDecode( const AStr : String) : string
1410: Function HTTPEncode( const AStr : String) : string
1411: Function Hypot( const X, Y : Extended) : Extended
1412: Function IBMax( n1, n2 : Integer) : Integer');
1413: Function IBMin( n1, n2 : Integer) : Integer');
1414: Function IBRandomString( iLength : Integer) : String');
1415: Function IBRandomInteger( iLow, iHigh : Integer) : Integer');
1416: Function IBStripString( st : String; CharsToStrip : String) : String');
1417: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String');
1418: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String');
1419: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String');
1420: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String');
1421: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string');
1422: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)');
1423: Function IconToBitmap( Ico : HICON) : TBitmap
1424: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1425: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1426: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean)
1427: function IdentToColor(const Ident: string; var Color: Longint): Boolean)
1428: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1429: Function IdGetDefaultCharSet : TIdCharSet
1430: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1431: Function IdPorts2 : TStringList
1432: Function IdToMib( const Id : string) : string
```

```
1433: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1434: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1435: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
1436: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1437: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean) : Boolean;
1438: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1439: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer) : TDateTime
1440: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1441: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1442: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1443: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1444: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1445: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1446: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1447: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1448: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1449: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1450: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1451: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1452: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1453: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1454: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1455: Function IncludeTrailingBackslash( S : string) : string
1456: function IncludeTrailingBackslash(const S: string): string)
1457: Function IncludeTrailingPathDelimiter( const APath : string) : string
1458: Function IncludeTrailingPathDelimiter( S : string) : string
1459: function IncludeTrailingPathDelimiter(const S: string): string)
1460: Function IncludeTrailingSlash( const APath : string) : string
1461: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliSeconds : Int64) : TDateTime
1462: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64) : TDateTime
1463: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer) : TDateTime
1464: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1465: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64) : TDateTime
1466: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer) : TDateTime
1467: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer) : TDateTime
1468: Function IndexOf( AClass : TClass) : Integer
1469: Function IndexOf( AComponent : TComponent) : Integer
1470: Function IndexOf( AObject : TObject) : Integer
1471: Function INDEXOF( const ANAME : String) : INTEGER
1472: Function IndexOf( const DisplayName : string) : Integer
1473: Function IndexOf( const Item : TBookmarkStr) : Integer
1474: Function IndexOf( const S : WideString) : Integer
1475: Function IndexOf( const View : TJclFileMappingView) : Integer
1476: Function INDEXOF( FIELD : TFIELD) : INTEGER
1477: Function IndexOf( ID : LCID) : Integer
1478: Function INDEXOF( ITEM : TMENUITEM) : INTEGER
1479: Function IndexOf( Value : TListItem) : Integer
1480: Function IndexOf( Value : TTreeNode) : Integer
1481: function IndexOf(const S: string): Integer;
1482: Function IndexOfName( const Name : WideString) : Integer
1483: function IndexOfName(Name: string): Integer;
1484: Function IndexOfObject( AObject : TObject) : Integer
1485: function IndexofObject(AObject:tObject):Integer
1486: Function IndexOfTabAt( X, Y : Integer) : Integer
1487: Function IndexStr( const AText : string; const AValues : array of string) : Integer
1488: Function IndexText( const AText : string; const AValues : array of string) : Integer
1489: Function IndexOfInteger( AList : TStringList; Value : Variant) : Integer');
1490: Function IndexOfFloat( AList : TStringList; Value : Variant) : Integer');
1491: Function IndexOfDate( AList : TStringList; Value : Variant) : Integer');
1492: Function IndexOfString( AList : TStringList; Value : Variant) : Integer');
1493: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1494: Function IndyGetHostName : string
1495: Function IndyInterlockedDecrement( var I : Integer) : Integer
1496: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1497: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
1498: Function IndyInterlockedIncrement( var I : Integer) : Integer
1499: Function IndyLowerCase( const A1 : string) : string
1500: Function IndyStrToBool( const AString : String) : Boolean
1501: Function IndyUpperCase( const A1 : string) : string
1502: Function InitCommonControl( CC : Integer) : Boolean
1503: Function InitTempPath : string
1504: Function InMainThread : boolean
1505: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1506: Function Input: Text)
1507: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1508: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1509: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1510: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1511: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1512: Function InquireSignal( RtlSigNum : Integer) : TSignalState
1513: Function InRangeR( const A, Min, Max : Double) : Boolean
1514: function Insert( Index : Integer) : TCollectionItem
1515: Function Insert( Index : Integer) : TComboExItem
1516: Function Insert( Index : Integer) : THeaderSection
1517: Function Insert( Index : Integer) : TListItem
1518: Function Insert( Index : Integer) : TStatusPanel
```

```
1519: Function Insert( Index : Integer) : TWorkArea
1520: Function Insert( Index : LongInt; const Text : string) : LongInt
1521: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1522: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1523: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1524: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1525: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
1526: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1527: Function Instance : Longint
1528: function InstanceSize: Longint
1529: Function Int(e : Extended) : Extended;
1530: function Int64ToStr(i: Int64): String;
1531: Function IntegerToBcd( const AValue : Integer) : TBcd
1532: Function Intensity( const Color32 : TColor32) : Integer;
1533: Function Intensity( const R, G, B : Single) : Single;
1534: Function InterestPayment( const Rate : Extended; Period, NPeriods : Integer; const PresentValue,
      FutureValue : Extended; PaymentTime : TPaymentTime) : Extended
1535: Function InterestRate( NPeriods : Integer; const Payment, PresentValue, FutureValue : Extended;
      PaymentTime : TPaymentTime) : Extended
1536: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1537: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1538: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1539: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1540: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1541: Function IntMibToStr( const Value : string) : string
1542: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1543: Function IntToBin( Value : cardinal) : string
1544: Function IntToHex( Value : Integer; Digits : Integer) : string;
1545: function IntToHex(a: integer; b: integer): string;
1546: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1547: function IntToHex64(Value: Int64; Digits: Integer): string)
1548: Function IntTo3Str( Value : Longint; separator: string) : string');
1549: Function inttobool( aInt : LongInt) : Boolean');
1550: function IntToStr(i: Int64): String;
1551: Function IntToStr64(Value: Int64): string)
1552: function IOResult: Integer
1553: Function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1554: Function IsAccel(VK: Word; const Str: string): Boolean
1555: Function IsAddressInNetwork( Address : String) : Boolean
1556: Function IsAdministrator : Boolean
1557: Function IsAlias( const Name : string) : Boolean
1558: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1559: Function IsASCII( const AByte : Byte) : Boolean;
1560: Function IsASCIILDH( const AByte : Byte) : Boolean;
1561: Function IsAssembly(const FileName: string): Boolean;
1562: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1563: Function IsBinary(const AChar : Char) : Boolean
1564: function IsConsole: Boolean)
1565: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1566: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1567: Function IsDelphiDesignMode : boolean
1568: Function IsDelphiRunning : boolean
1569: Function IsDFAState : boolean
1570: Function IsDirectory( const FileName : string) : Boolean
1571: Function IsDomain( const S : String) : Boolean
1572: function IsDragObject(Sender: TObject): Boolean;
1573: Function IsEditing : Boolean
1574: Function ISEMPTY : BOOLEAN
1575: Function IsEqual( Value : TParameters) : Boolean
1576: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1577: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1578: Function IsFirstNode : Boolean
1579: Function IsFloatZero( const X : Float) : Boolean
1580: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1581: Function IsFormOpen(const FormName: string): Boolean;
1582: Function IsFQDN( const S : String) : Boolean
1583: Function IsGrayScale : Boolean
1584: Function IsHex( AChar : Char) : Boolean;
1585: Function IsHexString(const AString: string): Boolean;
1586: Function IsHostname( const S : String) : Boolean
1587: Function IsInfinite( const AValue : Double) : Boolean
1588: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1589: Function IsInternet: boolean;
1590: Function IsLeadChar( ACh : Char) : Boolean
1591: Function IsLeapYear( Year : Word) : Boolean
1592: function IsLeapYear(Year: Word): Boolean)
1593: function IsLibrary: Boolean)
1594: Function ISLINE : BOOLEAN
1595: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1596: Function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1597: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1598: Function IsMainAppWindow( Wnd : HWND) : Boolean
1599: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1600: function IsMemoryManagerSet: Boolean)
1601: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1602: function IsMultiThread: Boolean)
```

```
1603: Function IsNumeric( AChar : Char) : Boolean;
1604: Function IsNumeric2( const AString : string) : Boolean;
1605: Function IsOctal( AChar : Char) : Boolean;
1606: Function IsOctalString(const AString: string) : Boolean;
1607: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1608: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1609: Function IsPM( const AValue : TDateTime) : Boolean
1610: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1611: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1612: Function IsPrimeRM( N : Cardinal) : Boolean
1613: Function IsPrimeTD( N : Cardinal) : Boolean
1614: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1615: Function ISqrt( const I : Smallint) : Smallint
1616: Function IsReadOnly(const Filename: string): boolean;
1617: Function IsRectEmpty( const Rect : TRect) : Boolean
1618: function IsRectEmpty(const Rect: TRect): Boolean)
1619: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1620: Function ISRIGHTTOLEFT : BOOLEAN
1621: function IsRightToLeft: Boolean
1622: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1623: Function ISSEQUENCED : BOOLEAN
1624: Function IsSystemModule( const Module : HMODULE) : Boolean
1625: Function IsSystemResourcesMeterPresent : Boolean
1626: Function IsToday( const AValue : TDateTime) : Boolean
1627: function IsToday(const AValue: TDateTime): Boolean;
1628: Function IsTopDomain( const AStr : string) : Boolean
1629: Function IsUTF8LeadByte( Lead : Char) : Boolean
1630: Function IsUTF8String( const s : UTF8String) : Boolean
1631: Function IsUTF8TrailByte( Lead : Char) : Boolean
1632: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1633: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1634: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1635: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1636: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1637: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1638: Function IsValidIdent( Ident : string) : Boolean
1639: function IsValidIdent(const Ident: string; AllowDots: Boolean): Boolean)
1640: Function IsValidIP( const S : String) : Boolean
1641: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean
1642: Function IsVariantManagerSet: Boolean;'); //deprecated;
1643: Function IsVirtualPcGuest : Boolean;
1644: Function IsVmWareGuest : Boolean;
1645: Function IsVCLControl(Handle: HWnd): Boolean;
1646: Function IsWhiteString( const AStr : String) : Boolean
1647: Function IsWindowResponding( Wnd : HWND; Timeout : Integer) : Boolean
1648: Function IsWoW64: boolean;
1649: Function IsWin64: boolean;
1650: Function IsWow64String(var s: string): Boolean;
1651: Function IsWin64String(var s: string): Boolean;
1652: Function isPowerof2(num: int64): boolean;
1653: Function powerOf2(exponent: integer): int64;
1654: function IsZero(const A: Extended; Epsilon: Extended): Boolean'); //overload;
1655: function IsZero1(const A: Double; Epsilon: Double): Boolean'); //overload;
1656: function IsZero2(const A: Single; Epsilon: Single): Boolean'); //overload;
1657: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1658: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1659: Function ItemRect( Index : Integer) : TRect
1660: function ITEMRECT(INDEX:INTEGER):TRECT
1661: Function ItemWidth( Index : Integer) : Integer
1662: Function JosephusG(n,k: integer; var graphout: string): integer;
1663: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1664: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;');
1665: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;');
1666: Function KeepAlive : Boolean
1667: Function KeysToShiftState(Keys: Word): TShiftState;');
1668: Function KeyDataToShiftState(KeyData: Longint): TShiftState;');
1669: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;');
1670: Function KeyboardStateToShiftState: TShiftState; overload;');
1671: Function Languages : TLanguages
1672: Function Last : TClass
1673: Function Last : TComponent
1674: Function Last : TObject
1675: Function LastDelimiter( Delimiters, S : string) : Integer
1676: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1677: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1678: Function Latitude2WGS84(lat: double): double;
1679: Function LCM(m,n:longint):longint;
1680: Function LCMJ( const X, Y : Cardinal) : Cardinal
1681: Function Ldexp( const X : Extended; const P : Integer) : Extended
1682: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1683: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1684: function Length: Integer;
1685: function lengthmp3(mp3path: string):integer;
1686: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1687: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1688: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
```

```
1689: function LineStart(Buffer, BufPos: PChar): PChar
1690: function LineStart(Buffer, BufPos: PChar): PChar)
1691: function ListSeparator: char;
1692: function Ln(x: Extended): Extended;
1693: Function LnXP1( const X : Extended) : Extended
1694: function Lo(vdat: word): byte;
1695: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1696: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1697: Function LoadFileAsString( const FileName : string) : string
1698: Function LoadFromFile( const FileName : string) : TBitmapLoader
1699: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1700: Function LoadPackage(const Name: string): HMODULE
1701: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1702: Function LoadStr( Ident : Integer) : string
1703: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1704: Function LoadWideStr( Ident : Integer) : WideString
1705: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1706: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1707: Function LockServer( fLock : LongBool) : HResult
1708: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1709: Function Log( const X : Extended) : Extended
1710: Function Log10( const X : Extended) : Extended
1711: Function Log2( const X : Extended) : Extended
1712: function LogBase10(X: Float): Float;
1713: Function LogBase2(X: Float): Float;
1714: Function LogBaseN(Base, X: Float): Float;
1715: Function LogN( const Base, X : Extended) : Extended
1716: Function LogOffOS : Boolean
1717: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean');
1718: Function LoginDialogEx(const ADatabaseName:string;var AUserName,
      APassword:string;NameReadOnly:Boolean):Boolean;
1719: Function LongDateFormat: string;
1720: function LongTimeFormat: string;
1721: Function LongWordToFourChar( ACardinal : LongWord) : string
1722: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1723: Function LookupName( const name : string) : TInAddr
1724: Function LookupService( const service : string) : Integer
1725: function Low: Int64;
1726: Function LowerCase( S : string) : string
1727: Function Lowercase(s : AnyString) : AnyString;
1728: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1729: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1730: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1731: function MainInstance: longword
1732: function MainThreadID: longword
1733: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1734: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;');
1735: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1736: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1737: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1738: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1739: function MakeLong(A, B: Word): Longint)
1740: Function MakeTempFilename( const APath : String) : string
1741: Function MakeValidFileName( const Str : string) : string
1742: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1743: function MakeWord(A, B: Byte): Word)
1744: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1745: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1746: Function MapValues( Mapping : string; Value : string) : string
1747: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1748: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1749: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1750: Function MaskGetFldSeparator( const EditMask : string) : Integer
1751: Function MaskGetMaskBlank( const EditMask : string) : Char
1752: Function MaskGetMaskSave( const EditMask : string) : Boolean
1753: Function MaskIntlLiteralToChar( IChar : Char) : Char
1754: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1755: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1756: Function MaskString( Mask, Value : String) : String
1757: Function Match( const sString : string) : TniRegularExpressionMatchResul
1758: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1759: Function Matches( const Filename : string) : Boolean
1760: Function MatchesMask( const Filename, Mask : string) : Boolean
1761: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1762: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1763: Function Max( AValueOne, AValueTwo : Integer) : Integer
1764: function Max(const x,y: Integer): Integer;
1765: Function Max1( const B1, B2 : Shortint) : Shortint;
1766: Function Max2( const B1, B2 : Smallint) : Smallint;
1767: Function Max3( const B1, B2 : Word) : Word;
1768: function Max3(const x,y,z: Integer): Integer;
1769: Function Max4( const B1, B2 : Integer) : Integer;
1770: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1771: Function Max6( const B1, B2 : Int64) : Int64;
1772: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1773: Function MaxFloat( const X, Y : Float) : Float
```

```
1774: Function MaxFloatArray( const B : TDynFloatArray) : Float
1775: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1776: function MaxIntValue(const Data: array of Integer):Integer)
1777: Function MaxJ( const B1, B2 : Byte) : Byte;
1778: function MaxPath: string;
1779: function MaxValue(const Data: array of Double): Double)
1780: Function MaxCalc( const Formula : string) : Extended');  //math expression parser
1781: Procedure MaxCalcF( const Formula : string);   //out to console memo2
1782: function MD5(const fileName: string): string;
1783: Function Mean( const Data : array of Double) : Extended
1784: Function Median( const X : TDynFloatArray) : Float
1785: Function Memory : Pointer
1786: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1787: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1788: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1789: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1790: Function MessageDlg1(const Msg: string; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx:Longint;
      DefaultButton:TMsgDlgBtn) : Integer;
1791: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1792: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
1793: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1794: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1795: Function MibToId( Mib : string) : string
1796: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1797: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1798: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1799: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut');
1800: Procedure GetMidiOutputs( const List : TStrings)');
1801: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData');
1802: Function MIDINoteToStr( Note : TMIDINote) : string');
1803: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut');
1804: Procedure GetMidiOutputs( const List : TStrings)');
1805: Procedure MidiOutCheck( Code : MMResult)');
1806: Procedure MidiInCheck( Code : MMResult)');
1807: Function MilliSecondOf( const AValue : TDateTime) : Word
1808: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1809: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1810: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1811: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1812: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1813: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
1814: Function MilliSecondOfTheYear( const AValue : TDateTime) : Int64
1815: Function MilliSecondsBetween( const ANow, AThen : TDateTime) : Int64
1816: Function MilliSecondSpan( const ANow, AThen : TDateTime) : Double
1817: Function Min( AValueOne, AValueTwo : Integer) : Integer
1818: Function Min1( const B1, B2 : Shortint) : Shortint;
1819: Function Min2( const B1, B2 : Smallint) : Smallint;
1820: Function Min3( const B1, B2 : Word) : Word;
1821: Function Min4( const B1, B2 : Integer) : Integer;
1822: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1823: Function Min6( const B1, B2 : Int64) : Int64;
1824: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1825: Function MinClientRect : TRect;
1826: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1827: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1828: Function MinFloat( const X, Y : Float) : Float
1829: Function MinFloatArray( const B : TDynFloatArray) : Float
1830: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1831: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1832: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1833: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1834: Function MinIntValue( const Data : array of Integer) : Integer
1835: function MinIntValue(const Data: array of Integer):Integer)
1836: Function MinJ( const B1, B2 : Byte) : Byte;
1837: Function MinuteOf( const AValue : TDateTime) : Word
1838: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1839: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1840: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1841: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1842: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1843: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1844: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1845: Function MinValue( const Data : array of Double) : Double
1846: function MinValue(const Data: array of Double): Double)
1847: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1848: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1849: Function ModFloat( const X, Y : Float) : Float
1850: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1851: Function Modify( const Key : string; Value : Integer) : Boolean
1852: Function ModuleCacheID : Cardinal
1853: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1854: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
```

```
1855: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1856: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1857: Function MonthOf( const AValue : TDateTime) : Word
1858: Function MonthOfTheYear( const AValue : TDateTime) : Word
1859: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1860: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1861: Function MonthStr( DateTime : TDateTime) : string
1862: Function MouseCoord( X, Y : Integer) : TGridCoord
1863: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1864: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1865: Function MoveNext : Boolean
1866: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1867: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1868: Function Name : string
1869: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1870: function NetworkVolume(DriveChar: Char): string
1871: Function NEWBOTTOMLINE : INTEGER
1872: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1873: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1874: Function NEWLINE : TMENUITEM
1875: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1876: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1877: Function NEWPOPUPMENU( OWNER : TCOMPONENT; const ANAME : String; ALIGNMENT : TPOPUPALIGNMENT; AUTOPOPUP :
      BOOLEAN; const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1878: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1879: Function NEWSUBMENU( const ACAPTION : String; HCTX : WORD; const ANAME : String; ITEMS : array of
      TMenuItem; AENABLED : BOOLEAN) : TMENUITEM
1880: Function NEWTOPLINE : INTEGER
1881: Function Next : TIdAuthWhatsNext
1882: Function NextCharIndex( S : String; Index : Integer) : Integer
1883: Function NextRecordSet : TCustomSQLDataSet
1884: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1885: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1886: Function NextToken : Char
1887: Function nextToken : WideString
1888: function NextToken:Char
1889: Function Norm( const Data : array of Double) : Extended
1890: Function NormalizeAngle( const Angle : Extended) : Extended
1891: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
1892: Function NormalizeRect( const Rect : TRect) : TRect
1893: function NormalizeRect(const Rect: TRect): TRect;
1894: Function Now : TDateTime
1895: function Now2: tDateTime
1896: Function NumProcessThreads : integer');
1897: Function NumThreadCount : integer');
1898: Function NthDayOfWeek( const AValue : TDateTime) : Word
1899: Function NtProductType : TNtProductType
1900: Function NtProductTypeString : string
1901: function Null: Variant;
1902: Function NullPoint : TPoint
1903: Function NullRect : TRect
1904: Function Null2Blank(aString:String):String;
1905: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
1906: Function NumIP : integer
1907: function Odd(x: Longint): boolean;
1908: Function OffsetFromUTC : TDateTime
1909: Function OffsetPoint( const P, Offset : TPoint) : TPoint
1910: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
1911: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
1912: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
1913: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
1914: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
1915: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
1916: function OpenBit:Integer
1917: Function OpenDatabase : TDatabase
1918: Function OpenDatabase( const DatabaseName : string) : TDatabase
1919: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
1920: Function OpenObject( Value : PChar) : Boolean;
1921: Function OpenObject1( Value : string) : Boolean;
1922: Function OpenSession( const SessionName : string) : TSession
1923: Function OpenVolume( const Drive : Char) : THandle
1924: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
1925: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
1926: Function OrdToBinary( const Value : Byte) : string;
1927: Function OrdToBinary1( const Value : Shortint) : string;
1928: Function OrdToBinary2( const Value : Smallint) : string;
1929: Function OrdToBinary3( const Value : Word) : string;
1930: Function OrdToBinary4( const Value : Integer) : string;
1931: Function OrdToBinary5( const Value : Cardinal) : string;
1932: Function OrdToBinary6( const Value : Int64) : string;
1933: Function OSCheck( RetVal : Boolean) : Boolean
1934: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
```

```
1935: Function OSIdentToString( const OSIdent : DWORD) : string
1936: Function Output: Text)
1937: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
1938: Function Owner : TCustomListView
1939: function Owner : TPersistent
1940: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
1941: Function PadL( pStr : String; pLth : integer) : String
1942: Function Padl(s : AnyString;I : longInt) : AnyString;
1943: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
1944: Function PadR( pStr : String; pLth : integer) : String
1945: Function Padr(s : AnyString;I : longInt) : AnyString;
1946: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
1947: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
1948: Function Padz(s : AnyString;I : longInt) : AnyString;
1949: Function PaethPredictor( a, b, c : Byte) : Byte
1950: Function PARAMBYNAME( const VALUE : String) : TPARAM
1951: Function ParamByName( const Value : WideString) : TParameter
1952: Function ParamCount: Integer
1953: Function ParamsEncode( const ASrc : string) : string
1954: function ParamStr(Index: Integer): string)
1955: Function ParseDate( const DateStr : string) : TDateTime
1956: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
1957: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
1958: Function PathAddExtension( const Path, Extension : string) : string
1959: Function PathAddSeparator( const Path : string) : string
1960: Function PathAppend( const Path, Append : string) : string
1961: Function PathBuildRoot( const Drive : Byte) : string
1962: Function PathCanonicalize( const Path : string) : string
1963: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
1964: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
1965: Function PathCompactPath1(const Canvas:TCanvas;const Path:string;const Width:Integer;CmpFmt:TCompactPath):
      string;
1966: Function PathEncode( const ASrc : string) : string
1967: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
1968: Function PathExtractFileNameNoExt( const Path : string) : string
1969: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
1970: Function PathGetDepth( const Path : string) : Integer
1971: Function PathGetLongName( const Path : string) : string
1972: Function PathGetLongName2( Path : string) : string
1973: Function PathGetShortName( const Path : string) : string
1974: Function PathIsAbsolute( const Path : string) : Boolean
1975: Function PathIsChild( const Path, Base : AnsiString) : Boolean
1976: Function PathIsDiskDevice( const Path : string) : Boolean
1977: Function PathIsUNC( const Path : string) : Boolean
1978: Function PathRemoveExtension( const Path : string) : string
1979: Function PathRemoveSeparator( const Path : string) : string
1980: Function Payment(Rate : Extended; NPeriods : Integer; const PresentValue, FutureValue : Extended;
      PaymentTime : TPaymentTime) : Extended
1981: Function Peek : Pointer
1982: Function Peek : TObject
1983: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
1984: Function PeriodPayment( const Rate : Extended; Period, NPeriods : Integer; const PresentValue, FutureValue
      : Extended; PaymentTime : TPaymentTime) : Extended
1985: function Permutation(npr, k: integer): extended;
1986: function PermutationInt(npr, k: integer): Int64;
1987: Function PermutationJ( N, R : Cardinal) : Float
1988: Function Pi : Extended;
1989: Function PiE : Extended;
1990: Function PixelsToDialogsX( const Pixels : Word) : Word
1991: Function PixelsToDialogsY( const Pixels : Word) : Word
1992: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
1993: Function Point( X, Y : Integer) : TPoint
1994: function Point(X, Y: Integer): TPoint)
1995: Function PointAssign( const X, Y : Integer) : TPoint
1996: Function PointDist( const P1, P2 : TPoint) : Double;
1997: function PointDist(const P1,P2: TFloatPoint): Double;
1998: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
1999: function PointDist2(const P1,P2: TPoint): Double;
2000: Function PointEqual( const P1, P2 : TPoint) : Boolean
2001: Function PointIsNull( const P : TPoint) : Boolean
2002: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2003: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2004: Function Pop : Pointer
2005: Function Pop : TObject
2006: Function PopnStdDev( const Data : array of Double) : Extended
2007: Function PopnVariance( const Data : array of Double) : Extended
2008: Function PopulationVariance( const X : TDynFloatArray) : Float
2009: function Pos(SubStr, S: AnyString): Longint;
2010: Function PosEqual( const Rect : TRect) : Boolean
2011: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2012: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2013: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2014: Function Post1( AURL : string; const ASource : TStrings) : string;
2015: Function Post2( AURL : string; const ASource : TStream) : string;
2016: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream) : string;
2017: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
```

```
2018: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2019: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2020: Function Power( const Base, Exponent : Extended) : Extended
2021: Function PowerBig(aval, n:integer): string;
2022: Function PowerIntJ( const X : Float; N : Integer) : Float;
2023: Function PowerJ( const Base, Exponent : Float) : Float;
2024: Function PowerOffOS : Boolean
2025: Function PreformatDateString( Ps : string) : string
2026: Function PresentValue( const Rate : Extended; NPeriods : Integer; const Payment, FutureValue : Extended;
      PaymentTime : TPaymentTime) : Extended
2027: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2028: Function Printer : TPrinter
2029: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2030: Function ProcessResponse : TIdHTTPWhatsNext
2031: Function ProduceContent : string
2032: Function ProduceContentFromStream( Stream : TStream) : string
2033: Function ProduceContentFromString( const S : string) : string
2034: Function ProgIDToClassID(const ProgID: string): TGUID;
2035: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2036: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2037: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
      const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2038: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string;const
      ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2039: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2040: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2041: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2042: Function Push( AItem : Pointer) : Pointer
2043: Function Push( AObject : TObject) : TObject
2044: Function Put1( AURL : string; const ASource : TStream) : string;
2045: Function Pythagoras( const X, Y : Extended) : Extended
2046: Function queryDLLInterface( var queryList : TStringList) : TStringList
2047: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2048: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2049: Function queryPerformanceCounter2(mse: int64): int64;');
2050: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;');
2051: //Function QueryPerformanceFrequency(mse: int64): boolean;');
2052: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;');
2053: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;');
2054: Procedure QueryPerformanceCounter1(var aC: Int64);');
2055: Function QueryPerformanceFrequency1(var freq: int64): boolean;');
2056: Function Quote( const ACommand : String) : SmallInt
2057: Function QuotedStr( S : string) : string
2058: Function RadToCycle( const Radians : Extended) : Extended
2059: Function RadToDeg( const Radians : Extended) : Extended
2060: Function RadToDeg( const Value : Extended) : Extended;
2061: Function RadToDeg1( const Value : Double) : Double;
2062: Function RadToDeg2( const Value : Single) : Single;
2063: Function RadToGrad( const Radians : Extended) : Extended
2064: Function RadToGrad( const Value : Extended) : Extended;
2065: Function RadToGrad1( const Value : Double) : Double;
2066: Function RadToGrad2( const Value : Single) : Single;
2067: Function RandG( Mean, StdDev : Extended) : Extended
2068: function Random(const ARange: Integer): Integer;
2069: function random2(a: integer): double
2070: function RandomE: Extended;
2071: function RandomF: Extended;
2072: Function RandomFrom( const AValues : array of string) : string;
2073: Function RandomRange( const AFrom, ATo : Integer) : Integer
2074: function randSeed: longint
2075: Function RawToDataColumn( ACol : Integer) : Integer
2076: Function Read : Char
2077: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2078: function Read(Buffer:String;Count:LongInt):LongInt
2079: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2080: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2081: Function ReadCardinal( const AConvert : boolean) : Cardinal
2082: Function ReadChar : Char
2083: Function ReadClient( var Buffer, Count : Integer) : Integer
2084: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2085: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2086: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2087: Function ReadFromStack(const ARaiseExceptionIfDisconnected:Bool;ATimeout:Integer;const
      ARaiseExceptionOnTimeout: Boolean):Integer
2088: Function ReadInteger( const AConvert : boolean) : Integer
2089: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2090: Function ReadLn : string
2091: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2092: function Readln(question: string): string;
2093: Function ReadLnWait( AFailCount : Integer) : string
2094: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;');
2095: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;');
2096: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2097: Function ReadString( const ABytes : Integer) : string
2098: Function ReadString( const Section, Ident, Default : string) : string
2099: Function ReadString( Count : Integer) : string
```

```
2100: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2101: Function ReadTimeStampCounter : Int64
2102: Function RebootOS : Boolean
2103: Function Receive( ATimeOut : Integer) : TReplyStatus
2104: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2105: Function ReceiveLength : Integer
2106: Function ReceiveText : string
2107: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal');
2108: Function ReceiveSerialText: string');
2109: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2110: Function RecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word) : TDateTime
2111: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2112: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2113: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2114: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2115: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2116: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2117: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2118: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2119: Function Reconcile( const Results : OleVariant) : Boolean
2120: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2121: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2122: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;');
2123: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2124: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2125: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2126: Function RectCenter( const R : TRect) : TPoint
2127: Function RectEqual( const R1, R2 : TRect) : Boolean
2128: Function RectHeight( const R : TRect) : Integer
2129: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2130: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2131: Function RectIntersection( const R1, R2 : TRect) : TRect
2132: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2133: Function RectIsEmpty( const R : TRect) : Boolean
2134: Function RectIsNull( const R : TRect) : Boolean
2135: Function RectIsSquare( const R : TRect) : Boolean
2136: Function RectIsValid( const R : TRect) : Boolean
2137: Function RectsAreValid( R : array of TRect) : Boolean
2138: Function RectUnion( const R1, R2 : TRect) : TRect
2139: Function RectWidth( const R : TRect) : Integer
2140: Function RedComponent( const Color32 : TColor32) : Integer
2141: Function Refresh : Boolean
2142: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2143: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2144: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2145: Function RegisterConversionType(const AFamily:TConvFamil;const ADescript:string;const AFact:Double):
      TConvType
2146: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2147: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2148: Function ReleaseHandle : HBITMAP
2149: Function ReleaseHandle : HENHMETAFILE
2150: Function ReleaseHandle : HICON
2151: Function ReleasePalette : HPALETTE
2152: Function RemainderFloat( const X, Y : Float) : Float
2153: Function Remove( AClass : TClass) : Integer
2154: Function Remove( AComponent : TComponent) : Integer
2155: Function Remove( AItem : Integer) : Integer
2156: Function Remove( AItem : Pointer) : Pointer
2157: Function Remove( AItem : TObject) : TObject
2158: Function Remove( AObject : TObject) : Integer
2159: Function RemoveBackslash( const PathName : string) : string
2160: Function RemoveDF( aString : String) : String');  //removes thousand separator
2161: Function RemoveDir( Dir : string) : Boolean
2162: function RemoveDir(const Dir: string): Boolean)
2163: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2164: Function RemoveFileExt( const FileName : string) : string
2165: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2166: Function RenameFile( OldName, NewName : string) : Boolean
2167: function RenameFile(const OldName: string; const NewName: string): Boolean)
2168: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2169: Function ReplaceText( const AText, AFromText, AToText : string) : string
2170: Function Replicate(c : char;I : longInt) : String;
2171: Function Request : TWebRequest
2172: Function ResemblesText( const AText, AOther : string) : Boolean
2173: Function Reset : Boolean
2174: function Reset2(mypath: string):string;
2175: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2176: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2177: Function Response : TWebResponse
2178: Function ResumeSupported : Boolean
2179: Function RETHINKHOTKEYS : BOOLEAN
2180: Function RETHINKLINES : BOOLEAN
2181: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2182: Function RetrieveCurrentDir : string
2183: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
```

```
2184: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2185: Function RetrieveMailBoxSize : integer
2186: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2187: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2188: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2189: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean');
2190: Function ReverseBits( Value : Byte) : Byte;
2191: Function ReverseBits1( Value : Shortint) : Shortint;
2192: Function ReverseBits2( Value : Smallint) : Smallint;
2193: Function ReverseBits3( Value : Word) : Word;
2194: Function ReverseBits4( Value : Cardinal) : Cardinal;
2195: Function ReverseBits4( Value : Integer) : Integer;
2196: Function ReverseBits5( Value : Int64) : Int64;
2197: Function ReverseBytes( Value : Word) : Word;
2198: Function ReverseBytes1( Value : Smallint) : Smallint;
2199: Function ReverseBytes2( Value : Integer) : Integer;
2200: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2201: Function ReverseBytes4( Value : Int64) : Int64;
2202: Function ReverseString( const AText : string) : string
2203: Function ReverseDNSLookup(const IPAddress: String; const DNSServer: String; Timeout, Retries: Integer; var
      HostName: String): Boolean;
2204: Function Revert : HResult
2205: Function RGB(R,G,B: Byte): TColor;
2206: Function RGB2BGR( const Color : TColor) : TColor
2207: Function RGB2TColor( R, G, B : Byte) : TColor
2208: Function RGBToWebColorName( RGB : Integer) : string
2209: Function RGBToWebColorStr( RGB : Integer) : string
2210: Function RightStr( const AStr : String; Len : Integer) : String
2211: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2212: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2213: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2214: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
2215: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2216: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2217: Function Round(e : Extended) : Longint;
2218: Function Round64(e: extended): Int64;
2219: Function RoundAt( const Value : string; Position : SmallInt) : string
2220: Function RoundFrequency( const Frequency : Integer) : Integer
2221: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2222: Function RoundPoint( const X, Y : Double) : TPoint
2223: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2224: Function RowCount : Integer
2225: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2226: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2227: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2228: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2229: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2230: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2231: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2232: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2233: Function S_AddBackSlash( const ADirName : string) : string
2234: Function S_AllTrim( const cStr : string) : string
2235: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2236: Function S_Cut( const cStr : string; const iLen : integer) : string
2237: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2238: Function S_DirExists( const ADir : string) : Boolean
2239: Function S_Empty( const cStr : string) : boolean
2240: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2241: Function S_LargeFontsActive : Boolean
2242: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2243: Function S_LTrim( const cStr : string) : string
2244: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2245: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2246: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2247: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2248: Function S_RTrim( const cStr : string) : string
2249: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2250: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2251: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2252: Function S_Space( const iLen : integer) : String
2253: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2254: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2255: Function S_StrCRC32( const Text : string) : LongWORD
2256: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2257: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2258: Function S_StringtoUTF_8( const AString : string) : string
2259: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2260: function S_StrToReal(const cStr: string; var R: Double): Boolean
2261: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2262: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2263: Function S_UTF_8ToString( const AString : string) : string
2264: Function S_WBox( const AText : string) : integer
2265: Function SameDate( const A, B : TDateTime) : Boolean
2266: function SameDate(const A, B: TDateTime): Boolean;
2267: Function SameDateTime( const A, B : TDateTime) : Boolean
2268: function SameDateTime(const A, B: TDateTime): Boolean;
```

```
2269: Function SameFileName( S1, S2 : string) : Boolean
2270: Function SameText( S1, S2 : string) : Boolean
2271: function SameText(const S1: string; const S2: string): Boolean)
2272: Function SameTime( const A, B : TDateTime) : Boolean
2273: function SameTime(const A, B: TDateTime): Boolean;
2274: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean'); //overload;
2275: function SameValue1(const A, B: Double; Epsilon: Double): Boolean'); //overload;
2276: function SameValue2(const A, B: Single; Epsilon: Single): Boolean'); //overload;
2277: Function SampleVariance( const X : TDynFloatArray) : Float
2278: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2279: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2280: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2281: Function SaveToFile( const AFileName : TFileName) : Boolean
2282: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2283: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;');
2284: Function ScanF(const aformat: String; const args: array of const): string;
2285: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2286: Function SearchBuf( Buf : PChar; BufLen : Integer; SelStart, SelLength : Integer; SearchString : String;
      Options : TStringSearchOptions) : PCha
2287: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2288: function SearchRecattr: integer;
2289: function SearchRecExcludeAttr: integer;
2290: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2291: function SearchRecname: string;
2292: function SearchRecsize: integer;
2293: function SearchRecTime: integer;
2294: Function Sec( const X : Extended) : Extended
2295: Function Secant( const X : Extended) : Extended
2296: Function SecH( const X : Extended) : Extended
2297: Function SecondOf( const AValue : TDateTime) : Word
2298: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2299: Function SecondOfTheHour( const AValue : TDateTime) : Word
2300: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2301: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2302: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2303: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2304: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2305: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2306: Function SectionExists( const Section : string) : Boolean
2307: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2308: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2309: function Seek(Offset:LongInt;Origin:Word):LongInt
2310: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2311: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2312: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean');
2313: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2314: Function SendBuf( var Buf, Count : Integer) : Integer
2315: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2316: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2317: Function SendKey( AppName : string; Key : Char) : Boolean
2318: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2319: Function SendStream( AStream : TStream) : Boolean
2320: Function SendStreamThenDrop( AStream : TStream) : Boolean
2321: Function SendText( const S : string) : Integer
2322: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal');
2323: Function SendSerialText(Data: String): cardinal');
2324: Function Sent : Boolean
2325: Function ServicesFilePath: string
2326: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2327: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2328: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2329: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2330: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2331: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2332: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2333: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2334: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2335: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2336: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2337: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2338: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2339: Function SetCurrentDir( Dir : string) : Boolean
2340: function SetCurrentDir(const Dir: string): Boolean)
2341: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2342: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2343: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2344: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2345: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2346: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2347: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2348: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2349: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2350: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2351: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
```

```
2352: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2353: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2354: Function SetLocalTime( Value : TDateTime) : boolean
2355: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2356: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2357: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2358: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2359: Function SetSize( libNewSize : Longint) : HResult
2360: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2361: Function Sgn( const X : Extended) : Integer
2362: function SHA1(const fileName: string): string;
2363: function SHA256(astr: string; amode: char): string)');
2364: function SHA512(astr: string; amode: char): string)');
2365: Function ShareMemoryManager : Boolean
2366: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2367: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2368: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2369: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2370: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2371: function ShortDateFormat: string;
2372: Function ShortenString( const DC : HDC; const S : WideString; const Width : Integer; const RTL : Boolean;
      EllipsisWidth : Integer) : WideString
2373: function ShortTimeFormat: string;
2374: function SHOWMODAL:INTEGER
2375: function ShowWindow(C1: HWND; C2: integer): boolean;
2376: Function ShutDownOS : Boolean
2377: Function Signe( const X, Y : Extended) : Extended
2378: Function Sign( const X : Extended) : Integer');
2379: Function Sin(e : Extended) : Extended;
2380: Function sinc( const x : Double) : Double
2381: Function SinJ( X : Float) : Float
2382: Function Size( const AFileName : String) : Integer
2383: function SizeOf: Longint;
2384: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2385: function SlashSep(const Path, S: String): String
2386: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2387: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD');
2388: Function SmallPoint(X, Y: Integer): TSmallPoint)
2389: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2390: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2391: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2392: Function SoundexProc( const AText, AOther : string) : Boolean
2393: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2394: Function SoundexWord( const AText : string) : Word
2395: Function SourcePos : Longint
2396: function SourcePos:LongInt
2397: Function Split0( Str : string; const substr : string) : TStringList
2398: Procedure SplitNameValue( const Line : string; var Name, Value : string)');
2399: Function SQLRequiresParams( const SQL : WideString) : Boolean
2400: Function Sqr(e : Extended) : Extended;
2401: Function Sqrt(e : Extended) : Extended;
2402: Function StartIP : String
2403: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean');
2404: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2405: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2406: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2407: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2408: Function StartOfAYear( const AYear : Word) : TDateTime
2409: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2410: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2411: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2412: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2413: Function StartsStr( const ASubText, AText : string) : Boolean
2414: Function StartsText( const ASubText, AText : string) : Boolean
2415: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2416: Function StartsWith( const str : string; const sub : string) : Boolean
2417: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2418: Function StatusString( StatusCode : Integer) : string
2419: Function StdDev( const Data : array of Double) : Extended
2420: Function Stop : Float
2421: Function StopCount( var Counter : TJclCounter) : Float
2422: Function StoreColumns : Boolean
2423: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2424: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2425: Function StrAlloc( Size : Cardinal) : PChar
2426: function StrAlloc(Size: Cardinal): PChar)
2427: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2428: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2429: Function StrBufSize( Str : PChar) : Cardinal
2430: function StrBufSize(const Str: PChar): Cardinal)
2431: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2432: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2433: Function StrCat( Dest : PChar; Source : PChar) : PChar
2434: function StrCat(Dest: PChar; const Source: PChar): PChar)
2435: Function StrCharLength( Str : PChar) : Integer
2436: Function StrComp( Str1, Str2 : PChar) : Integer
```

```
2437: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2438: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2439: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2440: Function Stream_to_AnsiString( Source : TStream) : ansistring
2441: Function Stream_to_Base64( Source : TStream) : ansistring
2442: Function Stream_to_decimalbytes( Source : TStream) : string
2443: Function Stream2WideString( oStream : TStream) : WideString
2444: Function StreamtoAnsiString( Source : TStream) : ansistring
2445: Function StreamToByte( Source : TStream) : string
2446: Function StreamToDecimalbytes( Source : TStream) : string
2447: Function StreamtoOrd( Source : TStream) : string
2448: Function StreamToString( Source : TStream) : string
2449: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2450: Function StrEmpty( const sString : string) : boolean
2451: Function StrEnd( Str : PChar) : PChar
2452: function StrEnd(const Str: PChar): PChar)
2453: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2454: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2455: Function StrGet(var S : String; I : Integer) : Char;
2456: Function StrGet2(S : String; I : Integer) : Char;
2457: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2458: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2459: Function StrHtmlDecode( const AStr : String) : String
2460: Function StrHtmlEncode( const AStr : String) : String
2461: Function StrToBytes(const Value: String): TBytes;
2462: Function StrIComp( Str1, Str2 : PChar) : Integer
2463: Function StringOfChar(c : char;I : longInt) : String;
2464: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2465: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2466: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2467: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2468: Function StringToBoolean( const Ps : string) : Boolean
2469: function StringToColor(const S: string): TColor)
2470: function StringToCursor(const S: string): TCursor;
2471: function StringToGUID(const S: string): TGUID)
2472: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2473: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2474: Function StringWidth( S : string) : Integer
2475: Function StrInternetToDateTime( Value : string) : TDateTime
2476: Function StrIsDateTime( const Ps : string) : Boolean
2477: Function StrIsFloatMoney( const Ps : string) : Boolean
2478: Function StrIsInteger( const S : string) : Boolean
2479: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2480: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2481: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2482: Function StrLen( Str : PChar) : Cardinal
2483: function StrLen(const Str: PChar): Cardinal)
2484: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2485: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2486: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2487: Function StrLower( Str : PChar) : PChar
2488: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2489: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2490: Function StrNew( Str : PChar) : PChar
2491: function StrNew(const Str: PChar): PChar)
2492: Function StrNextChar( Str : PChar) : PChar
2493: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2494: Function StrParse( var sString : string; const sDelimiters : string) : string;
2495: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2496: Function StrPas( Str : PChar) : string
2497: function StrPas(const Str: PChar): string)
2498: Function StrPCopy( Dest : PChar; Source : string) : PChar
2499: function StrPCopy(Dest: PChar; const Source: string): PChar)
2500: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2501: Function StrPos( Str1, Str2 : PChar) : PChar
2502: Function StrScan(const Str: PChar; Chr: Char): PChar)');
2503: Function StrRScan(const Str: PChar; Chr: Char): PChar)');
2504: Function StrToBcd( const AValue : string) : TBcd
2505: Function StrToBool( S : string) : Boolean
2506: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2507: Function StrToCard( const AStr : String) : Cardinal
2508: Function StrToConv( AText : string; out AType : TConvType) : Double
2509: Function StrToCurr( S : string) : Currency;
2510: function StrToCurr(const S: string): Currency)
2511: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2512: Function StrToDate( S : string) : TDateTime;
2513: function StrToDate(const s: string): TDateTime;
2514: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2515: Function StrToDateTime( S : string) : TDateTime;
2516: function StrToDateTime(const S: string): TDateTime)
2517: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2518: Function StrToDay( const ADay : string) : Byte
2519: Function StrToFloat( S : string) : Extended;
2520: function StrToFloat(s: String): Extended;
2521: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2522: function StrToFloatDef(const S: string; const Default: Extended): Extended)
```

```
2523: Function StrToFloat( S : string) : Extended;');
2524: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;');
2525: Function StrToFloatDef( S : string; Default : Extended) : Extended;');
2526: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;');
2527: Function StrToCurr( S : string) : Currency;');
2528: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;');
2529: Function StrToCurrDef( S : string; Default : Currency) : Currency;');
2530: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;');
2531: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;');
2532: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;');
2533: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;');
2534: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;');
2535: Function StrToDateTime( S : string) : TDateTime;');
2536: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;');
2537: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;');
2538: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2539: Function StrToInt( S : string) : Integer
2540: function StrToInt(s: String): Longint;
2541: Function StrToInt64( S : string) : Int64
2542: function StrToInt64(s: String): int64;
2543: Function StrToInt64Def( S : string; Default : Int64) : Int64
2544: Function StrToInt64Def(const S: string; const Default: Int64):Int64)
2545: Function StrToIntDef( S : string; Default : Integer) : Integer
2546: function StrToIntDef(const S: string; Default: Integer): Integer)
2547: function StrToIntDef(s: String; def: Longint): Longint;
2548: Function StrToMonth( const AMonth : string) : Byte
2549: Function StrToTime( S : string) : TDateTime;
2550: function StrToTime(const S: string): TDateTime)
2551: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2552: Function StrToWord( const Value : String) : Word
2553: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2554: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2555: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2556: Function StrUpper( Str : PChar) : PChar
2557: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2558: Function Sum( const Data : array of Double) : Extended
2559: Function SumFloatArray( const B : TDynFloatArray) : Float
2560: Function SumInt( const Data : array of Integer) : Integer
2561: Function SumOfSquares( const Data : array of Double) : Extended
2562: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2563: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2564: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2565: Function Supports( CursorOptions : TCursorOptions) : Boolean
2566: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2567: Function SwapWord(w : word): word)');
2568: Function SwapInt(i : integer): integer)');
2569: Function SwapLong(L : longint): longint)');
2570: Function Swap(i : integer): integer)');
2571: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2572: Function SyncTime : Boolean
2573: Function SysErrorMessage( ErrorCode : Integer) : string
2574: function SysErrorMessage(ErrorCode: Integer): string)
2575: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2576: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2577: Function SysStringLen(const S: WideString): Integer; stdcall;
2578: Function TabRect( Index : Integer) : TRect
2579: Function Tan( const X : Extended) : Extended
2580: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2581: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2582: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2583: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2584: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2585: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2586: Function TenToY( const Y : Float) : Float
2587: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2588: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2589: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2590: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2591: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2592: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2593: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2594: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2595: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2596: Function TestBits( const Value, Mask : Byte) : Boolean;
2597: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2598: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2599: Function TestBits3( const Value, Mask : Word) : Boolean;
2600: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2601: Function TestBits5( const Value, Mask : Integer) : Boolean;
2602: Function TestBits6( const Value, Mask : Int64) : Boolean;
```

```
2603: Function TestFDIVInstruction : Boolean
2604: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2605: Function TextExtent( const Text : string) : TSize
2606: function TextHeight(Text: string): Integer;
2607: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2608: Function TextStartsWith( const S, SubS : string) : Boolean
2609: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2610: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2611: function TextWidth(Text: string): Integer;
2612: Function ThreadCount : integer');
2613: function ThousandSeparator: char;
2614: Function Ticks : Cardinal
2615: Function Time : TDateTime
2616: function Time: TDateTime;
2617: function TimeGetTime: int64;
2618: Function TimeOf( const AValue : TDateTime) : TDateTime
2619: function TimeSeparator: char;
2620: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2621: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2622: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2623: Function TimeToStr( DateTime : TDateTime) : string;
2624: function TimeToStr(const DateTime: TDateTime): string;
2625: Function TimeZoneBias : TDateTime
2626: Function ToCommon( const AValue : Double) : Double
2627: function ToCommon(const AValue: Double): Double;
2628: Function Today : TDateTime
2629: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2630: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2631: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2632: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2633: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2634: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2635: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2636: function TokenComponentIdent:String
2637: Function TokenFloat : Extended
2638: function TokenFloat:Extended
2639: Function TokenInt : Longint
2640: function TokenInt:LongInt
2641: Function TokenString : string
2642: function TokenString:String
2643: Function TokenSymbolIs( const S : string) : Boolean
2644: function TokenSymbolIs(S:String):Boolean
2645: Function Tomorrow : TDateTime
2646: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2647: Function ToString : string
2648: Function TotalVariance( const Data : array of Double) : Extended
2649: Function Trace2( AURL : string) : string;
2650: Function TrackMenu( Button : TToolButton) : Boolean
2651: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2652: Function TranslateURI( const URI : string) : string
2653: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2654: Function TransparentStretchBlt( DstDC : HDC; DstX, DstY, DstW, DstH : Integer; SrcDC : HDC; SrcX, SrcY,
      SrcW, SrcH : Integer; MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2655: Function Trim( S : string) : string;
2656: Function Trim( S : WideString) : WideString;
2657: Function Trim(s : AnyString) : AnyString;
2658: Function TrimAllOf( ATrim, AText : String) : String
2659: Function TrimLeft( S : string) : string;
2660: Function TrimLeft( S : WideString) : WideString;
2661: function TrimLeft(const S: string): string)
2662: Function TrimRight( S : string) : string;
2663: Function TrimRight( S : WideString) : WideString;
2664: function TrimRight(const S: string): string)
2665: function TrueBoolStrs: array of string
2666: Function Trunc(e : Extended) : Longint;
2667: Function Trunc64(e: extended): Int64;
2668: Function TruncPower( const Base, Exponent : Float) : Float
2669: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2670: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2671: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2672: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2673: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2674: Function TryEncodeDateTime( const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond : Word; out
      AValue : TDateTime) : Boolean
2675: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2676: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out AValue:TDateTime):
      Boolean
2677: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2678: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2679: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2680: Function TryLock : Boolean
2681: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2682: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
      AMilliSecond : Word; out AResult : TDateTime) : Boolean
2683: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2684: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
```

```
2685: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2686: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2687: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2688: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2689: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2690: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2691: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2692: Function TwoToY( const Y : Float) : Float
2693: Function UCS4StringToWideString( const S : UCS4String) : WideString
2694: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2695: function Unassigned: Variant;
2696: Function UndoLastChange( FollowChange : Boolean) : Boolean
2697: function UniCodeToStr(Value: string): string;
2698: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2699: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2700: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2701: Function UnixPathToDosPath( const Path : string) : string
2702: Function UnixToDateTime( const AValue : Int64) : TDateTime
2703: function UnixToDateTime(U: Int64): TDateTime;
2704: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2705: Function UnlockResource( ResData : HGLOBAL) : LongBool
2706: Function UnlockVolume( var Handle : THandle) : Boolean
2707: Function UnMaskString( Mask, Value : String) : String
2708: function UpCase(ch : Char ) : Char;
2709: Function UpCaseFirst( const AStr : string) : string
2710: Function UpCaseFirstWord( const AStr : string) : string
2711: Function UpdateAction( Action : TBasicAction) : Boolean
2712: Function UpdateKind : TUpdateKind
2713: Function UPDATESTATUS : TUPDATESTATUS
2714: Function UpperCase( S : string) : string
2715: Function Uppercase(s : AnyString) : AnyString;
2716: Function URLDecode( ASrc : string) : string
2717: Function URLEncode( const ASrc : string) : string
2718: Function UseRightToLeftAlignment : Boolean
2719: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2720: Function UseRightToLeftReading : Boolean
2721: Function UTF8CharLength( Lead : Char) : Integer
2722: Function UTF8CharSize( Lead : Char) : Integer
2723: Function UTF8Decode( const S : UTF8String) : WideString
2724: Function UTF8Encode( const WS : WideString) : UTF8String
2725: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2726: Function Utf8ToAnsi( const S : UTF8String) : string
2727: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2728: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2729: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2730: Function ValidParentForm(control: TControl): TForm
2731: Function Value : Variant
2732: Function ValueExists( const Section, Ident : string) : Boolean
2733: Function ValueOf( const Key : string) : Integer
2734: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2735: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2736: Function VarArrayFromStrings( Strings : TStrings) : Variant
2737: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2738: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2739: Function VarFMTBcd : TVarType
2740: Function VarFMTBcdCreate1 : Variant;
2741: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2742: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2743: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2744: Function Variance( const Data : array of Double) : Extended
2745: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2746: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2747: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2748: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2749: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2750: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2751: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2752: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2753: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2754: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2755: Function VariantNeg( const V1 : Variant) : Variant
2756: Function VariantNot( const V1 : Variant) : Variant
2757: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2758: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2759: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2760: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2761: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2762: function VarIsEmpty(const V: Variant): Boolean;
2763: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2764: function VarIsNull(const V: Variant): Boolean;
2765: Function VarToBcd( const AValue : Variant) : TBcd
2766: function VarType(const V: Variant): TVarType;
2767: Function VarType( const V : Variant) : TVarType');
2768: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant');
2769: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;');
2770: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;');
```

```
2771: Function VarIsByRef( const V : Variant) : Boolean');
2772: Function VarIsEmpty( const V : Variant) : Boolean');
2773: Procedure VarCheckEmpty( const V : Variant)');
2774: Function VarIsNull( const V : Variant) : Boolean');
2775: Function VarIsClear( const V : Variant) : Boolean');
2776: Function VarIsCustom( const V : Variant) : Boolean');
2777: Function VarIsOrdinal( const V : Variant) : Boolean');
2778: Function VarIsFloat( const V : Variant) : Boolean');
2779: Function VarIsNumeric( const V : Variant) : Boolean');
2780: Function VarIsStr( const V : Variant) : Boolean');
2781: Function VarToStr( const V : Variant) : string');
2782: Function VarToStrDef( const V : Variant; const ADefault : string) : string');
2783: Function VarToWideStr( const V : Variant) : WideString');
2784: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString');
2785: Function VarToDateTime( const V : Variant) : TDateTime');
2786: Function VarFromDateTime( const DateTime : TDateTime) : Variant');
2787: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean');
2788: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant');
2789: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )');
2790: Function VarSameValue( const A, B : Variant) : Boolean');
2791: Function VarCompareValue( const A, B : Variant) : TVariantRelationship');
2792: Function VarIsEmptyParam( const V : Variant) : Boolean');
2793: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;');
2794: Function VarIsError1( const V : Variant) : Boolean;');
2795: Function VarAsError( AResult : HRESULT) : Variant');
2796: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)');
2797: Function VarIsArray( const A : Variant) : Boolean;');
2798: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;');
2799: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant');
2800: Function VarArrayOf( const Values : array of Variant) : Variant');
2801: Function VarArrayRef( const A : Variant) : Variant');
2802: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean');
2803: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean');
2804: Function VarArrayDimCount( const A : Variant) : Integer');
2805: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer');
2806: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer');
2807: Function VarArrayLock( const A : Variant) : ___Pointer');
2808: Procedure VarArrayUnlock( const A : Variant)');
2809: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant');
2810: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)');
2811: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)');
2812: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)');
2813: Function Unassigned : Variant');
2814: Function Null : Variant');
2815: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2816: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2817: Function VectorDot( const V1, V2 : TFloatPoint) : Double
2818: function VectorDot(const V1,V2: TFloatPoint): Double;
2819: Function VectorLengthSqr( const V : TFloatPoint) : Double
2820: function VectorLengthSqr(const V: TFloatPoint): Double;
2821: Function VectorMult( const V : TFloatPoint; const s : Double) : TFloatPoint
2822: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2823: Function VectorSubtract( const V1, V2 : TFloatPoint) : TFloatPoint
2824: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2825: Function Verify( AUserName : String) : String
2826: Function Versine( X : Float) : Float
2827: function VersionCheck: boolean;
2828: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string
2829: Function VersionLanguageName( const LangId : Word) : string
2830: Function VersionResourceAvailable( const FileName : string) : Boolean
2831: Function Visible : Boolean
2832: function VolumeID(DriveChar: Char): string
2833: Function WaitFor( const AString : string) : string
2834: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult
2835: Function WaitFor1 : TWaitResult;
2836: Function WaitForData( Timeout : Longint) : Boolean
2837: Function WebColorNameToColor( WebColorName : string) : TColor
2838: Function WebColorStrToColor( WebColor : string) : TColor
2839: Function WebColorToRGB( WebColor : Integer) : Integer
2840: Function wGet(aURL, afile: string): boolean;'
2841: Function WebGet(aURL, afile: string): boolean;'
2842: Function WebExists: boolean;');   //alias to isinternet
2843: Function WeekOf( const AValue : TDateTime) : Word
2844: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2845: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2846: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2847: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2848: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer
2849: Function WeeksInAYear( const AYear : Word) : Word
2850: Function WeeksInYear( const AValue : TDateTime) : Word
2851: Function WeekSpan( const ANow, AThen : TDateTime) : Double
2852: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString
2853: Function WideCat( const x, y : WideString) : WideString
2854: Function WideCompareStr( S1, S2 : WideString) : Integer
2855: function WideCompareStr(const S1: WideString; const S2: WideString): Integer)
2856: Function WideCompareText( S1, S2 : WideString) : Integer
```

```
2857: function WideCompareText(const S1: WideString; const S2: WideString): Integer)
2858: Function WideCopy( const src : WideString; index, count : Integer) : WideString
2859: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString
2860: Function WideEqual( const x, y : WideString) : Boolean
2861: function WideFormat(const Format: WideString; const Args: array of const): WideString)
2862: Function WideGreater( const x, y : WideString) : Boolean
2863: Function WideLength( const src : WideString) : Integer
2864: Function WideLess( const x, y : WideString) : Boolean
2865: Function WideLowerCase( S : WideString) : WideString
2866: function WideLowerCase(const S: WideString): WideString)
2867: Function WidePos( const src, sub : WideString) : Integer
2868: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString
2869: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString
2870: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString
2871: Function WideSameStr( S1, S2 : WideString) : Boolean
2872: function WideSameStr(const S1: WideString; const S2: WideString): Boolean)
2873: Function WideSameText( S1, S2 : WideString) : Boolean
2874: function WideSameText(const S1: WideString; const S2: WideString): Boolean)
2875: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring
2876: Function WideStringToUCS4String( const S : WideString) : UCS4String
2877: Function WideUpperCase( S : WideString) : WideString
2878: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean
2879: function Win32Check(RetVal: boolean): boolean)
2880: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
2881: Function Win32RestoreFile( const FileName : string) : Boolean
2882: Function Win32Type : TIdWin32Type
2883: Function WinColor( const Color32 : TColor32) : TColor
2884: function winexec(FileName: pchar; showCmd: integer): integer;
2885: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
2886: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
2887: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
2888: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
2889: Function WithinPastMilliSeconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64) : Boolean
2890: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
2891: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
2892: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
2893: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer) : Boolean
2894: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
2895: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD');
2896: Function WordToStr( const Value : Word) : String
2897: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint) : Boolean');
2898: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean');
2899: Procedure GetWordGridFormatValues( Proc : TGetStrProc)');
2900: Function WorkArea : Integer
2901: Function WrapText( Line : string; MaxCol : Integer) : string;
2902: Function WrapText( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
2903: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
2904: function Write(Buffer:String;Count:LongInt):LongInt
2905: Function WriteClient( var Buffer, Count : Integer) : Integer
2906: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
2907: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
2908: Function WriteString( const AString : string) : Boolean
2909: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
2910: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer');
2911: Function wsprintf( Output : PChar; Format : PChar) : Integer');
2912: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
2913: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
2914: Function XorDecode( const Key, Source : string) : string
2915: Function XorEncode( const Key, Source : string) : string
2916: Function XorString( const Key, Src : ShortString) : ShortString
2917: Function Yield : Bool');
2918: Function YearOf( const AValue : TDateTime) : Word
2919: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
2920: Function YearSpan( const ANow, AThen : TDateTime) : Double
2921: Function Yesterday : TDateTime
2922: Function YesNoDialog(const ACaption, AMsg: string): boolean;
2923: Function( const Name : string; Proc : TUserFunction)
2924: Function Special_Scholz from 3.8.5.0
2925: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
2926: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
2927: Function FloatToTime2Dec(value:Extended):Extended;
2928: Function MinToStd(value:Extended):Extended;
2929: Function MinToStdAsString(value:Extended):String;
2930: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
2931: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
2932: Function Round2Dec (zahl:Extended):Extended;
2933: Function GetAngle(x,y:Extended):Double;
2934: Function AddAngle(a1,a2:Double):Double;
2935:
2936: ****************************************************************
2937: unit uPSI_StText;
2938: ****************************************************************
2939: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean');
2940: Function TextFileSize( var F : TextFile) : LongInt');
2941: Function TextPos( var F : TextFile) : LongInt');
2942: Function TextFlush( var F : TextFile) : Boolean');
```

```
2943:
2944: ********************************************************************
2945: from  JvVCLUtils;
2946: ********************************************************************
2947: { Windows resources (bitmaps and icons) VCL-oriented routines }
2948: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
2949: procedure DrawBitmapRectTransparent(Dest: TCanvas; DstX, DstY: Integer;SrcRect: TRect; Bitmap: TBitmap;
     TransparentColor: TColor);
2950: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
     Bitmap: TBitmap; TransparentColor: TColor);
2951: function MakeBitmap(ResID: PChar): TBitmap;
2952: function MakeBitmapID(ResID: Word): TBitmap;
2953: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
2954: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
2955: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
2956: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
2957:   HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
2958: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
2959: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
2960: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
2961: {$IFDEF WIN32}
2962: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
2963:   X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
2964: {$ENDIF}
2965: function MakeIcon(ResID: PChar): TIcon;
2966: function MakeIconID(ResID: Word): TIcon;
2967: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
2968: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
2969: {$IFDEF WIN32}
2970: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
2971: {$ENDIF}
2972: { Service routines }
2973: procedure NotImplemented;
2974: procedure ResourceNotFound(ResID: PChar);
2975: function PointInRect(const P: TPoint; const R: TRect): Boolean;
2976: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
2977: function PaletteColor(Color: TColor): Longint;
2978: function WidthOf(R: TRect): Integer;
2979: function HeightOf(R: TRect): Integer;
2980: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
2981: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
2982: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
2983: procedure Delay(MSecs: Longint);
2984: procedure CenterControl(Control: TControl);
2985: Function PaletteEntries( Palette : HPALETTE) : Integer');
2986: Function WindowClassName( Wnd : HWND) : string');
2987: Function ScreenWorkArea : TRect');
2988: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)');
2989: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)');
2990: Procedure ActivateWindow( Wnd : HWND)');
2991: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)');
2992: Procedure CenterWindow( Wnd : HWND)');
2993: Procedure ShadeRect( DC : HDC; const Rect : TRect)');
2994: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)');
2995: Function DialogsToPixelsX( Dlgs : Word) : Word');
2996: Function DialogsToPixelsY( Dlgs : Word) : Word');
2997: Function PixelsToDialogsX( Pixs : Word) : Word');
2998: Function PixelsToDialogsY( Pixs : Word) : Word');
2999: {$IFDEF WIN32}
3000: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3001: function MakeVariant(const Values: array of Variant): Variant;
3002: {$ENDIF}
3003: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3004: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3005: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3006: {$IFDEF CBUILDER}
3007: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3008: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3009: {$ELSE}
3010: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3011: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3012: {$ENDIF CBUILDER}
3013: function IsForegroundTask: Boolean;
3014: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3015: function GetAveCharSize(Canvas: TCanvas): TPoint;
3016: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3017: procedure FreeUnusedOle;
3018: procedure Beep;
3019: function GetWindowsVersionJ: string;
3020: function LoadDLL(const LibName: string): THandle;
3021: function RegisterServer(const ModuleName: string): Boolean;
3022: {$IFNDEF WIN32}
3023: function IsLibrary: Boolean;
3024: {$ENDIF}
3025: { Gradient filling routine }
3026: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
```

```
3027: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
      TFillDirection; Colors: Byte);
3028: { String routines }
3029: function GetEnvVar(const VarName: string): string;
3030: function AnsiUpperFirstChar(const S: string): string;
3031: function StringToPChar(var S: string): PChar;
3032: function StrPAlloc(const S: string): PChar;
3033: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3034: function DropT(const S: string): string;
3035: { Memory routines }
3036: function AllocMemo(Size: Longint): Pointer;
3037: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3038: procedure FreeMemo(var fpBlock: Pointer);
3039: function GetMemoSize(fpBlock: Pointer): Longint;
3040: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3041: {$IFNDEF COMPILER5_UP}
3042: procedure FreeAndNil(var Obj);
3043: {$ENDIF}
3044: // from PNGLoader
3045: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3046: Procedure TransformRGB2LOCO( Image : TLinearBitmap)');
3047: Procedure TransformLOCO2RGB( Image : TLinearBitmap)');
3048: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)');
3049:
3050: //****************************************added  from jvjvclutils
3051: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3052: function ReplaceComponentReference(This, NewReference: TComponent; var VarReference: TComponent): Boolean;
3053: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3054: function IsPositiveResult(Value: TModalResult): Boolean;
3055: function IsNegativeResult(Value: TModalResult): Boolean;
3056: function IsAbortResult(const Value: TModalResult): Boolean;
3057: function StripAllFromResult(const Value: TModalResult): TModalResult;
3058: // returns either BrightColor or DarkColor depending on the luminance of AColor
3059: // This function gives the same result (AFAIK) as the function used in Windows to
3060: // calculate the desktop icon text color based on the desktop background color
3061: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3062: type
3063:   TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3064:
3065: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3066:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3067:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3068:   var LinkName: string; Scale: Integer = 100); overload;
3069: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3070:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3071:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3072:   var LinkName: string; Scale: Integer = 100); overload;
3073: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3074:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3075: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3076:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3077:   Scale: Integer = 100): string;
3078: function HTMLPlainText(const Text: string): string;
3079: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3080:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3081: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3082:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3083: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3084: function HTMLPrepareText(const Text: string): string;
3085:
3086:
3087: ****************************************** uPSI_JvAppUtils;
3088: Function GetDefaultSection( Component : TComponent) : string
3089: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3090: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3091: Function GetDefaultIniName : string
3092:  //CL.AddConstantN('OnGetDefaultIniName','TOnGetDefaultIniName').SetString();
3093: Function GetDefaultIniRegKey : string
3094: Function FindForm( FormClass : TFormClass) : TForm
3095: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3096: Function ShowDialog( FormClass : TFormClass) : Boolean
3097:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3098: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3099: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3100: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3101: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3102: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3103: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3104: Function StrToIniStr( const Str : string) : string
3105: Function IniStrToStr( const Str : string) : string
3106: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3107: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3108: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3109: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3110: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3111: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
```

```
3112: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3113: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3114: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3115: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3116: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3117: Procedure AppTaskbarIcons( AppOnly : Boolean)
3118: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3119: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3120: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3121: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3122: ****************************************** uPSI_JvDBUtils;
3123: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject');
3124: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3125: Procedure RefreshQuery( Query : TDataSet)
3126: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Boolean):Boolean
3127: Function DataSetSectionName( DataSet : TDataSet) : string
3128: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string)
3129: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const
      Section:string;RestoreVisible:Boolean)
3130: Function DataSetLocateThrough(DataSet: TDataSet; const KeyFields: string; const KeyValues: Variant;
      Options: TLocateOptions) : Boolean
3131: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3132: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3133: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3134: Function ConfirmDelete : Boolean
3135: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3136: Procedure CheckRequiredField( Field : TField)
3137: Procedure CheckRequiredFields( const Fields : array of TField)
3138: Function DateToSQL( Value : TDateTime) : string
3139: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3140: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3141: Function FormatSQLNumericRange( const FieldName: string; LowValue,HighValue,LowEmpty,
      HighEmpty:Double;Inclusive:Boolean) : string
3142: Function StrMaskSQL( const Value : string) : string
3143: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3144: Function FormatAnsiSQLCondition(const FieldName,Operator,Value:string;FieldType:TFieldType;Exact:Boolean):
      string
3145: Procedure _DBError( const Msg : string)
3146:  AddConstantN('TrueExpr','String').SetString( '0=0
3147:  AddConstantN('sdfStandard16','String').SetString( '''"''mm''/''dd''/''yyyy''"''
3148:  AddConstantN('sdfStandard32','String').SetString( ''''''dd/mm/yyyy''''''
3149:  AddConstantN('sdfOracle','String').SetString( '"TO_DATE(''"dd/mm/yyyy"'', ''DD/MM/YYYY'')"
3150:  AddConstantN('sdfInterbase','String').SetString( '"CAST(''"mm"/"dd"/"yyyy"'' AS DATE)"
3151:  AddConstantN('sdfMSSQL','String').SetString( '"CONVERT(datetime, ''"mm"/"dd"/"yyyy"'', 103)"
3152:  AddTypeS('Largeint', 'Longint
3153:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3154:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3155:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3156:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3157:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3158: ********************************************** uPSI_JvDBUtil;
3159: Procedure ExecuteSQLScript( Base : TDataBase; const Script : string; const Commit : TCommit; OnProgress :
      TOnProgress; const UserData : Integer)
3160: Function GetQueryResult( const DatabaseName, SQL : string) : Variant
3161: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant;
      const AResultName : string) : Variant
3162: //Function StrFieldDesc( Field : FLDDesc) : string
3163: Function Var2Type( V : Variant; const VarType : Integer) : Variant
3164: Procedure CopyRecord( DataSet : TDataSet)
3165: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string;
      MasterField : Word; ModOp, DelOp : RINTQual)
3166: Procedure AddMasterPassword( Table : TTable; pswd : string)
3167: Procedure PackTable( Table : TTable)
3168: Procedure PackEncryptedTable( Table : TTable; pswd : string)
3169: Function EncodeQuotes( const S : string) : string
3170: Function Cmp( const S1, S2 : string) : Boolean
3171: Function SubStr( const S : string; const Index : Integer; const Separator : string) : string
3172: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string) : string
3173: Function ReplaceString( S : string; const OldPattern, NewPattern : string) : string
3174: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer)
3175: ********************************************uPSI_JvJvBDEUtils;***************
3176: //JvBDEUtils
3177: Function CreateDbLocate : TJvLocateObject');
3178:  //Function CheckOpen( Status : DBIResult) : Boolean');
3179: Procedure FetchAllRecords( DataSet : TBDEDataSet)');
3180: Function TransActive( Database : TDatabase) : Boolean');
3181: Function AsyncQrySupported( Database : TDatabase) : Boolean');
3182: Function GetQuoteChar( Database : TDatabase) : string');
3183: Procedure ExecuteQuery( const DbName, QueryText : string)');
3184: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string)');
3185: Function FieldLogicMap( FldType : TFieldType) : Integer');
3186: Function FieldSubtypeMap( FldType : TFieldType) : Integer'); Value : string; Buffer : Pointer)');
3187: Function GetAliasPath( const AliasName : string) : string');
3188: Function IsDirectory( const DatabaseName : string) : Boolean');
3189: Function GetBdeDirectory : string');
3190: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean');
```

```
3191: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean');
3192: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean');
3193: Function DataSetRecNo( DataSet : TDataSet) : Longint');
3194: Function DataSetRecordCount( DataSet : TDataSet) : Longint');
3195: Function DataSetPositionStr( DataSet : TDataSet) : string');
3196: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)');
3197: Function CurrentRecordDeleted( DataSet : TBDEDataSet) : Boolean');
3198: Function IsFilterApplicable( DataSet : TDataSet) : Boolean');
3199: Function IsBookmarkStable( DataSet : TBDEDataSet) : Boolean');
3200: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)');
3201: Procedure RestoreIndex( Table : TTable)');
3202: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)');
3203: Procedure PackTable( Table : TTable)');
3204: Procedure ReindexTable( Table : TTable)');
3205: Procedure BdeFlushBuffers');
3206: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer) : Pointer');
3207: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)');
3208: Procedure DbNotSupported');
3209: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)');
3210: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3211: Procedure
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3212: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3213: ******************************************uPSI_JvDateUtil;
3214: function CurrentYear: Word;
3215: function IsLeapYear(AYear: Integer): Boolean;
3216: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3217: function FirstDayOfPrevMonth: TDateTime;
3218: function LastDayOfPrevMonth: TDateTime;
3219: function FirstDayOfNextMonth: TDateTime;
3220: function ExtractDay(ADate: TDateTime): Word;
3221: function ExtractMonth(ADate: TDateTime): Word;
3222: function ExtractYear(ADate: TDateTime): Word;
3223: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3224: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3225: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3226: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3227: function ValidDate(ADate: TDateTime): Boolean;
3228: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3229: function MonthsBetween(Date1, Date2: TDateTime): Double;
3230: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3231: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3232: function DaysBetween(Date1, Date2: TDateTime): Longint;
3233: { The same as previous but if Date2 < Date1 result = 0 }
3234: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3235: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3236: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3237: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3238: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3239: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3240: { String to date conversions }
3241: function GetDateOrder(const DateFormat: string): TDateOrder;
3242: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3243: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3244: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3245: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3246: function DefDateFormat(FourDigitYear: Boolean): string;
3247: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3248: ----------------------------------------------------------------------------
3249: ******************************** JvUtils;*******************************
3250: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3251: function GetWordOnPos(const S: string; const P: Integer): string;
3252: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3253: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3254: { SubStr returns substring from string, S, separated with Separator string}
3255: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3256: { SubStrEnd same to previous function but Index numerated from the end of string }
3257: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3258: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3259: function SubWord(P: PChar; var P2: PChar): string;
3260: { NumberByWord returns the text representation of
3261:   the number, N, in normal russian language. Was typed from Monitor magazine }
3262: function NumberByWord(const N: Longint): string;
3263: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3264:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3265: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3266: { GetXYByPos is same to previous function, but returns X position in line too}
3267: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3268: { ReplaceString searches for all substrings, OldPattern,in a string, S, and replaces them with NewPattern }
3269: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3270: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3271: function ConcatSep(const S, S2, Separator: string): string;
3272: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3273: function ConcatLeftSep(const S, S2, Separator: string): string;
```

```
3274: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3275: function MinimizeString(const S: string; const MaxLen: Integer): string;
3276: { Next 4 function for russian chars transliterating.
3277:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3278: procedure Dos2Win(var S: string);
3279: procedure Win2Dos(var S: string);
3280: function Dos2WinRes(const S: string): string;
3281: function Win2DosRes(const S: string): string;
3282: function Win2Koi(const S: string): string;
3283: { Spaces returns string consists on N space chars }
3284: function Spaces(const N: Integer): string;
3285: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3286: function AddSpaces(const S: string; const N: Integer): string;
3287: { function LastDate for russian users only } { returns date relative to current date: '' }
3288: function LastDate(const Dat: TDateTime): string;
3289: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3290: function CurrencyToStr(const Cur: currency): string;
3291: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3292: function Cmp(const S1, S2: string): Boolean;
3293: { StringCat add S2 string to S1 and returns this string }
3294: function StringCat(var S1: string; S2: string): string;
3295: { HasChar returns True, if Char, Ch, contains in string, S }
3296: function HasChar(const Ch: Char; const S: string): Boolean;
3297: function HasAnyChar(const Chars: string; const S: string): Boolean;
3298: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3299: function CountOfChar(const Ch: Char; const S: string): Integer;
3300: function DefStr(const S: string; Default: string): string;
3301: {**** files routines}
3302: { GetWinDir returns Windows folder name }
3303: function GetWinDir: TFileName;
3304: function GetSysDir: String;
3305: { GetTempDir returns Windows temporary folder name }
3306: function GetTempDir: string;
3307: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3308: function GenTempFileName(FileName: string): string;
3309: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3310: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3311: { ClearDir clears folder Dir }
3312: function ClearDir(const Dir: string): Boolean;
3313: { DeleteDir clears and than delete folder Dir }
3314: function DeleteDir(const Dir: string): Boolean;
3315: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3316: function FileEquMask(FileName, Mask: TFileName): Boolean;
3317: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3318:   Masks must be separated with comma (';') }
3319: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3320: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3321: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3322: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3323: { FileGetInfo fills SearchRec record for specified file attributes}
3324: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3325: { HasSubFolder returns True, if folder APath contains other folders }
3326: function HasSubFolder(APath: TFileName): Boolean;
3327: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3328: function IsEmptyFolder(APath: TFileName): Boolean;
3329: { AddSlash add slash Char to Dir parameter, if needed }
3330: procedure AddSlash(var Dir: TFileName);
3331: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3332: function AddSlash2(const Dir: TFileName): string;
3333: { AddPath returns FileName with Path, if FileName not contain any path }
3334: function AddPath(const FileName, Path: TFileName): TFileName;
3335: function AddPaths(const PathList, Path: string): string;
3336: function ParentPath(const Path: TFileName): TFileName;
3337: function FindInPath(const FileName, PathList: string): TFileName;
3338: function FindInPaths(const fileName,paths: String): String;
3339: {$IFNDEF BCB1}
3340: { BrowseForFolder displays Browse For Folder dialog }
3341: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3342: {$ENDIF BCB1}
3343: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean');
3344: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean');
3345: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3346: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3347:
3348: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3349: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3350: { HasParam returns True, if program running with specified parameter, Param }
3351: function HasParam(const Param: string): Boolean;
3352: function HasSwitch(const Param: string): Boolean;
3353: function Switch(const Param: string): string;
3354: { ExePath returns ExtractFilePath(ParamStr(0)) }
3355: function ExePath: TFileName;
3356: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3357: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
```

```
3358: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3359: {**** Graphic routines }
3360: { TTFontSelected returns True, if True Type font is selected in specified device context }
3361: function TTFontSelected(const DC: HDC): Boolean;
3362: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3363: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3364: {**** Windows routines }
3365: { SetWindowTop put window to top without recreating window }
3366: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3367: {**** other routines }
3368: { KeyPressed returns True, if Key VK is now pressed }
3369: function KeyPressed(VK: Integer): Boolean;
3370: procedure SwapInt(var Int1, Int2: Integer);
3371: function IntPower(Base, Exponent: Integer): Integer;
3372: function ChangeTopException(E: TObject): TObject;
3373: function StrToBool(const S: string): Boolean;
3374: {$IFNDEF COMPILER3_UP}
3375: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3376:   Length of MaxLen bytes. The compare operation is controlled by the
3377:   current Windows locale. The return value is the same as for CompareStr. }
3378: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3379: function AnsiStrIComp(S1, S2: PChar): Integer;
3380: {$ENDIF}
3381: function Var2Type(V: Variant; const VarType: Integer): Variant;
3382: function VarToInt(V: Variant): Integer;
3383: function VarToFloat(V: Variant): Double;
3384: { following functions are not documented because they are don't work properly , so don't use them }
3385: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3386: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3387: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3388: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3389: function GetParameter: string;
3390: function GetLongFileName(FileName: string): string;
3391: {* from  FileCtrl}
3392: function DirectoryExists(const Name: string): Boolean;
3393: procedure ForceDirectories(Dir: string);
3394: {# from  FileCtrl}
3395: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3396: function GetComputerID: string;
3397: function GetComputerName: string;
3398: {**** string routines }
3399: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 function }
3400: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3401: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3402:   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 }
3403: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3404: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3405: function CountOfLines(const S: string): Integer;
3406: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3407: procedure DeleteEmptyLines(Ss: TStrings);
3408: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3409:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3410: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3411: {**** files routines - }
3412: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3413:   Resource can be compressed using MS Compress program}
3414: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3415: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3416: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3417: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3418: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3419: { IniReadSection read section, Section, from ini-file,
3420:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3421:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3422: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3423: { LoadTextFile load text file, FileName, into string }
3424: function LoadTextFile(const FileName: TFileName): string;
3425: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3426: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3427: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3428: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3429: {$IFDEF COMPILER3_UP}
3430: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3431: function TargetFileName(const FileName: TFileName): TFileName;
3432: { return filename ShortCut linked to }
3433: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3434: {$ENDIF COMPILER3_UP}
3435: {**** Graphic routines - }
3436: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3437: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3438: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3439: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3440: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3441: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
```

```
{ RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
{ Cinema draws some visual effect }
procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
{ Roughed fills rect with special 3D pattern }
procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
{ 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 }
function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
{ TextWidth calculate text with for writing using standard desktop font }
function TextWidth(AStr: string): Integer;
{ DefineCursor load cursor from resource, and return available cursor number, assigned to it }
function DefineCursor(Identifer: PChar): TCursor;
{**** 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 : Integer; State : TDragState; var
Accept : Boolean; Sorted : Boolean)');
 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
 AddConstantN('MaxInitStrNum','LongInt').SetInt( 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');
```

```
3524: Function JvAnsiStrSplitStrings( const InString : AnsiString; const SplitChar, QuoteChar : AnsiChar;
      OutStrings : TStrings) : Integer');
3525: Function JvAnsiStrStrip( S : AnsiString) : AnsiString');
3526: Function JvStrStrip( S : string) : string');
3527: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString');
3528: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString');
3529: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString');
3530: Function StrEatWhiteSpace( const S : string) : string');
3531: Function HexToAscii( const S : AnsiString) : AnsiString');
3532: Function AsciiToHex( const S : AnsiString) : AnsiString');
3533: Function StripQuotes( const S1 : AnsiString) : AnsiString');
3534: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean');
3535: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean');
3536: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean');
3537: Function HexPCharToInt( S1 : PAnsiChar) : Integer');
3538: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean');
3539: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString');
3540: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean');
3541: Function JvValidIdentifier( S1 : String) : Boolean');
3542: Function JvEndChar( X : AnsiChar) : Boolean');
3543: Procedure JvGetToken( S1, S2 : PAnsiChar)');
3544: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean');
3545: Function IsKeyword( S1 : PAnsiChar) : Boolean');
3546: Function JvValidVarReference( S1 : PAnsiChar) : Boolean');
3547: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean');
3548: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)');
3549: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);');
3550: Procedure JvEatWhitespaceChars1( S1 : PWideChar);');
3551: Function GetTokenCount : Integer');
3552: Procedure ResetTokenCount');
3553: end;
3554:
3555: ******************************** JvStrUtil /  JvStrUtils;***************************
3556: function FindNotBlankCharPos(const S: string): Integer;
3557: function AnsiChangeCase(const S: string): string;
3558: function GetWordOnPos(const S: string; const P: Integer): string;
3559: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3560: function Cmp(const S1, S2: string): Boolean;
3561: { Spaces returns string consists on N space chars }
3562: function Spaces(const N: Integer): string;
3563: { HasChar returns True, if char, Ch, contains in string, S }
3564: function HasChar(const Ch: Char; const S: string): Boolean;
3565: function HasAnyChar(const Chars: string; const S: string): Boolean;
3566: { SubStr returns substring from string, S, separated with Separator string}
3567: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3568: { SubStrEnd same to previous function but Index numerated from the end of string }
3569: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3570: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3571: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3572: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3573: { GetXYByPos is same to previous function, but returns X position in line too}
3574: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3575: { AddSlash returns string with added slash char to Dir parameter, if needed }
3576: function AddSlash2(const Dir: TFileName): string;
3577: { AddPath returns FileName with Path, if FileName not contain any path }
3578: function AddPath(const FileName, Path: TFileName): TFileName;
3579: { ExePath returns ExtractFilePath(ParamStr(0)) }
3580: function ExePath: TFileName;
3581: function LoadTextFile(const FileName: TFileName): string;
3582: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3583: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3584: function ConcatSep(const S, S2, Separator: string): string;
3585: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3586: function FileEquMask(FileName, Mask: TFileName): Boolean;
3587: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3588:   Masks must be separated with comma (';') }
3589: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3590: function StringEndsWith(const Str, SubStr: string): Boolean;
3591: function ExtractFilePath2(const FileName: string): string;
3592: function StrToOem(const AnsiStr: string): string;
3593: { StrToOem translates a string from the Windows character set into the OEM character set. }
3594: function OemToAnsiStr(const OemStr: string): string;
3595: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
3596: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
3597: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
3598: function ReplaceStr(const S, Srch, Replace: string): string;
3599: { Returns string with every occurrence of Srch string replaced with Replace string. }
3600: function DelSpace(const S: string): string;
3601: { DelSpace return a string with all white spaces removed. }
3602: function DelChars(const S: string; Chr: Char): string;
3603: { DelChars return a string with all Chr characters removed. }
3604: function DelBSpace(const S: string): string;
3605: { DelBSpace trims leading spaces from the given string. }
3606: function DelESpace(const S: string): string;
3607: { DelESpace trims trailing spaces from the given string. }
3608: function DelRSpace(const S: string): string;
```

```
3609: { DelRSpace trims leading and trailing spaces from the given string. }
3610: function DelSpace1(const S: string): string;
3611: { DelSpace1 return a string with all non-single white spaces removed. }
3612: function Tab2Space(const S: string; Numb: Byte): string;
3613: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
3614: function NPos(const C: string; S: string; N: Integer): Integer;
3615: { NPos searches for a N-th position of substring C in a given string. }
3616: function MakeStr(C: Char; N: Integer): string;
3617: function MS(C: Char; N: Integer): string;
3618: { MakeStr return a string of length N filled with character C. }
3619: function AddChar(C: Char; const S: string; N: Integer): string;
3620: { AddChar return a string left-padded to length N with characters C. }
3621: function AddCharR(C: Char; const S: string; N: Integer): string;
3622: { AddCharR return a string right-padded to length N with characters C. }
3623: function LeftStr(const S: string; N: Integer): string;
3624: { LeftStr return a string right-padded to length N with blanks. }
3625: function RightStr(const S: string; N: Integer): string;
3626: { RightStr return a string left-padded to length N with blanks. }
3627: function CenterStr(const S: string; Len: Integer): string;
3628: { CenterStr centers the characters in the string based upon the Len specified. }
3629: function CompStr(const S1, S2: string): Integer;
3630: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
3631: function CompText(const S1, S2: string): Integer;
3632: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
3633: function Copy2Symb(const S: string; Symb: Char): string;
3634: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
3635: function Copy2SymbDel(var S: string; Symb: Char): string;
3636: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
3637: function Copy2Space(const S: string): string;
3638: { Copy2Symb returns a substring of a string S from begining to first white space. }
3639: function Copy2SpaceDel(var S: string): string;
3640: { Copy2SpaceDel returns a substring of a string S from begining to first
3641:   white space and removes this substring from S. }
3642: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
3643: { Returns string, with the first letter of each word in uppercase,
3644:   all other letters in lowercase. Words are delimited by WordDelims. }
3645: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
3646: { WordCount given a set of word delimiters, returns number of words in S. }
3647: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
3648: { Given a set of word delimiters, returns start position of N'th word in S. }
3649: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
3650: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
3651: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
3652: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
3653:   delimiters, return the N'th word in S. }
3654: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
3655: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
3656: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
3657: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
3658: function QuotedString(const S: string; Quote: Char): string;
3659: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
3660: function ExtractQuotedString(const S: string; Quote: Char): string;
3661: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
3662:    and reduces pairs of Quote characters within the quoted string to a single character. }
3663: function FindPart(const HelpWilds, InputStr: string): Integer;
3664: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
3665: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
3666: { IsWild compares InputString with WildCard string and returns True if corresponds. }
3667: function XorString(const Key, Src: ShortString): ShortString;
3668: function XorEncode(const Key, Source: string): string;
3669: function XorDecode(const Key, Source: string): string;
3670: { ** Command line routines ** }
3671: {$IFNDEF COMPILER4_UP}
3672: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
3673: {$ENDIF}
3674: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
3675: { ** Numeric string handling routines ** }
3676: function Numb2USA(const S: string): string;
3677: { Numb2USA converts numeric string S to USA-format. }
3678: function Dec2Hex(N: Longint; A: Byte): string;
3679: function D2H(N: Longint; A: Byte): string;
3680: { Dec2Hex converts the given value to a hexadecimal string representation
3681:   with the minimum number of digits (A) specified. }
3682: function Hex2Dec(const S: string): Longint;
3683: function H2D(const S: string): Longint;
3684: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
3685: function Dec2Numb(N: Longint; A, B: Byte): string;
3686: { Dec2Numb converts the given value to a string representation with the
3687:   base equal to B and with the minimum number of digits (A) specified. }
3688: function Numb2Dec(S: string; B: Byte): Longint;
3689: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
3690: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
3691: { IntToBin converts given value to a bin string representation with min number of digits specified. }
3692: function IntToRoman(Value: Longint): string;
3693: { IntToRoman converts the given value to a roman numeric string representation. }
3694: function RomanToInt(const S: string): Longint;
```

```
3695: { RomanToInt converts the given string to an integer value. If the string
3696:   doesn't contain a valid roman numeric value, the 0 value is returned. }
3697: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
3698: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
3699: ********************************* JvFileUtil;*********************************
3700: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
3701: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
3702: procedure MoveFile(const FileName, DestName: TFileName);
3703: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
3704: {$IFDEF COMPILER4_UP}
3705: function GetFileSize(const FileName: string): Int64;
3706: {$ELSE}
3707: function GetFileSize(const FileName: string): Longint;
3708: {$ENDIF}
3709: function FileDateTime(const FileName: string): TDateTime;
3710: function HasAttr(const FileName: string; Attr: Integer): Boolean;
3711: function DeleteFiles(const FileMask: string): Boolean;
3712: function DeleteFilesEx(const FileMasks: array of string): Boolean;
3713: function ClearDir(const Path: string; Delete: Boolean): Boolean;
3714: function NormalDir(const DirName: string): string;
3715: function RemoveBackSlash(const DirName: string): string;
3716: function ValidFileName(const FileName: string): Boolean;
3717: function DirExists(Name: string): Boolean;
3718: procedure ForceDirectories(Dir: string);
3719: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
3720:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
3721: {$IFDEF COMPILER4_UP}
3722: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
3723: {$ENDIF}
3724: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
3725: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
3726: {$IFDEF COMPILER4_UP}
3727: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
3728: {$ENDIF}
3729: function GetTempDir: string;
3730: function GetWindowsDir: string;
3731: function GetSystemDir: string;
3732: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
3733: {$IFDEF WIN32}
3734: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
3735: function ShortToLongFileName(const ShortName: string): string;
3736: function ShortToLongPath(const ShortName: string): string;
3737: function LongToShortFileName(const LongName: string): string;
3738: function LongToShortPath(const LongName: string): string;
3739: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
3740: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
3741: {$ENDIF WIN32}
3742: {$IFNDEF COMPILER3_UP}
3743: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
3744: {$ENDIF}
3745: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm');
3746: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl');
3747: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl');
3748: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)');
3749:
3750: //*************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
3751:  Procedure VariantClear( var V : Variant)');
3752:  Procedure VariantArrayRedim( var V : Variant; High : Integer)');
3753:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)');
3754:  Procedure VariantCpy( const src : Variant; var dst : Variant)');
3755:  Procedure VariantAdd( const src : Variant; var dst : Variant)');
3756:  Procedure VariantSub( const src : Variant; var dst : Variant)');
3757:  Procedure VariantMul( const src : Variant; var dst : Variant)');
3758:  Procedure VariantDiv( const src : Variant; var dst : Variant)');
3759:  Procedure VariantMod( const src : Variant; var dst : Variant)');
3760:  Procedure VariantAnd( const src : Variant; var dst : Variant)');
3761:  Procedure VariantOr( const src : Variant; var dst : Variant)');
3762:  Procedure VariantXor( const src : Variant; var dst : Variant)');
3763:  Procedure VariantShl( const src : Variant; var dst : Variant)');
3764:  Procedure VariantShr( const src : Variant; var dst : Variant)');
3765:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant');
3766:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant');
3767:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant');
3768:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant');
3769:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant');
3770:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant');
3771:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant');
3772:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant');
3773:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant');
3774:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant');
3775:  Function VariantNot( const V1 : Variant) : Variant');
3776:  Function VariantNeg( const V1 : Variant) : Variant');
3777:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;');
3778:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;');
3779:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;');
```

```
3780:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;');
3781:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;');
3782:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);');
3783:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);');
3784:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);');
3785:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);');
3786:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);');
3787: end;
3788:
3789: ************************************unit uPSI_JvgUtils;************************************
3790: function IsEven(I: Integer): Boolean;
3791: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
3792: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
3793: procedure SwapInt(var I1, I2: Integer);
3794: function Spaces(Count: Integer): string;
3795: function DupStr(const Str: string; Count: Integer): string;
3796: function DupChar(C: Char; Count: Integer): string;
3797: procedure Msg(const AMsg: string);
3798: function RectW(R: TRect): Integer;
3799: function RectH(R: TRect): Integer;
3800: function IncColor(AColor: Longint; AOffset: Byte): Longint;
3801: function DecColor(AColor: Longint; AOffset: Byte): Longint;
3802: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
3803: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
3804:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
3805: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
3806: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
3807:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor, HighlightColor,
      ShadowColor: TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
3808: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
3809: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
3810:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
3811: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
3812: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
3813: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
3814:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
3815:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
3816:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
3817: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
3818:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
3819:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
3820:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
3821: procedure BringParentWindowToTop(Wnd: TWinControl);
3822: function GetParentForm(Control: TControl): TForm;
3823: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
3824: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
3825: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
3826: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
3827: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
3828: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
3829: function CalcMathString(AExpression: string): Single;
3830: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
3831: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
3832: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
3833: procedure TypeStringOnKeyboard(const S: string);
3834: function NextStringGridCell( Grid: TStringGrid ): Boolean;
3835: procedure DrawTextExtAligned(Canvas:TCanvas;const
      Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
3836: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
3837: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
3838: function ComponentToString(Component: TComponent): string;
3839: procedure StringToComponent(Component: TComponent; const Value: string);
3840: function PlayWaveResource(const ResName: string): Boolean;
3841: function UserName: string;
3842: function ComputerName: string;
3843: function CreateIniFileName: string;
3844: function ExpandString(const Str: string; Len: Integer): string;
3845: function Transliterate(const Str: string; RusToLat: Boolean): string;
3846: function IsSmallFonts: Boolean;
3847: function SystemColorDepth: Integer;
3848: function GetFileTypeJ(const FileName: string): TglFileType;
3849: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
3850: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
3851:
3852: { ************************************Utility routines of unit classes}
3853: function LineStart(Buffer, BufPos: PChar): PChar');
3854: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
3855:   'Strings: TStrings): Integer');
3856:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat');
3857:  Procedure RegisterClass( AClass : TPersistentClass)');
3858:  Procedure RegisterClasses( AClasses : array of TPersistentClass)');
3859:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)');
3860:  Procedure UnRegisterClass( AClass : TPersistentClass)');
3861:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)');
3862:  Procedure UnRegisterModuleClasses( Module : HMODULE)');
3863:  Function FindGlobalComponent( const Name : string) : TComponent');
```

```
3864:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean');
3865:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean');
3866:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean');
3867:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent');
3868:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent');
3869:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent');
3870:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)');
3871:  Procedure GlobalFixupReferences');
3872:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)');
3873:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)');
3874:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)');
3875:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)');
3876:  Procedure RemoveFixups( Instance : TPersistent)');
3877:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent');
3878:  Procedure BeginGlobalLoading');
3879:  Procedure NotifyGlobalLoading');
3880:  Procedure EndGlobalLoading');
3881:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;');
3882:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;');
3883:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)');
3884:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer');
3885:  Procedure FreeObjectInstance( ObjectInstance : Pointer)');
3886: // Function AllocateHWnd( Method : TWndMethod) : HWND');
3887:  Procedure DeallocateHWnd( Wnd : HWND)');
3888:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean');
3889: ************************************unit uPSI_SqlTimSt and DB;**********************************
3890: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);');
3891: Function VarSQLTimeStampCreate3: Variant;');
3892: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;');
3893: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;');
3894: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;');
3895: Function VarSQLTimeStamp : TVarType');
3896: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;');
3897: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp'); //beta
3898: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp'); //beta
3899: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp');
3900: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string');
3901: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer');
3902: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp');
3903: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime');
3904: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean');
3905: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp');
3906: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)');
3907: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;');
3908: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;');
3909:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)');
3910: Procedure DatabaseError( const Message : WideString; Component : TComponent)');
3911: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)');
3912: Procedure DisposeMem( var Buffer, Size : Integer)');
3913: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean');
3914: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField');
3915: Function VarTypeToDataType( VarType : Integer) : TFieldType');
3916: ****************************************unit JvStrings;*************************************
3917: {template functions}
3918: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
3919: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
3920: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
3921: function RemoveMasterBlocks(const SourceStr: string): string;
3922: function RemoveFields(const SourceStr: string): string;
3923: {http functions}
3924: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
3925: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
3926: {set functions}
3927: procedure SplitSet(AText: string; AList: TStringList);
3928: function JoinSet(AList: TStringList): string;
3929: function FirstOfSet(const AText: string): string;
3930: function LastOfSet(const AText: string): string;
3931: function CountOfSet(const AText: string): Integer;
3932: function SetRotateRight(const AText: string): string;
3933: function SetRotateLeft(const AText: string): string;
3934: function SetPick(const AText: string; AIndex: Integer): string;
3935: function SetSort(const AText: string): string;
3936: function SetUnion(const Set1, Set2: string): string;
3937: function SetIntersect(const Set1, Set2: string): string;
3938: function SetExclude(const Set1, Set2: string): string;
3939: {replace any <,> etc by &lt; &gt;}
3940: function XMLSafe(const AText: string): string;
3941: {simple hash, Result can be used in Encrypt}
3942: function Hash(const AText: string): Integer;
3943: { Base64 encode and decode a string }
3944: function B64Encode(const S: AnsiString): AnsiString;
3945: function B64Decode(const S: AnsiString): AnsiString;
3946: {Basic encryption from a Borland Example}
3947: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
3948: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
3949: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
```

```
3950: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
3951: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
3952: procedure CSVToTags(Src, Dst: TStringList);
3953: // converts a csv list to a tagged string list
3954: procedure TagsToCSV(Src, Dst: TStringList);
3955: // converts a tagged string list to a csv list
3956: // only fieldnames from the first record are scanned ib the other records
3957: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
3958: {selects akey=avalue from Src and returns recordset in Dst}
3959: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
3960: {filters Src for akey=avalue}
3961: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
3962: {orders a tagged Src list by akey}
3963: function PosStr(const FindString, SourceString: string;
3964:   StartPos: Integer = 1): Integer;
3965: { PosStr searches the first occurrence of a substring FindString in a string
3966:   given by SourceString with case sensitivity (upper and lower case characters
3967:   are differed). This function returns the index value of the first character
3968:   of a specified substring from which it occurs in a given string starting with
3969:   StartPos character index. If a specified substring is not found Q_PosStr
3970:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
3971: function PosStrLast(const FindString, SourceString: string): Integer;
3972: {finds the last occurance}
3973: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
3974: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
3975: { PosText searches the first occurrence of a substring FindString in a string
3976:   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
3977: function PosTextLast(const FindString, SourceString: string): Integer;
3978: {finds the last occurance}
3979: function NameValuesToXML(const AText: string): string;
3980: {$IFDEF MSWINDOWS}
3981: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
3982: {$ENDIF MSWINDOWS}
3983: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
3984: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
3985: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
3986: procedure SaveString(const AFile, AText: string);
3987: Procedure SaveStringasFile( const AFile, AText : string)');
3988: function LoadStringJ(const AFile: string): string;
3989: Function LoadStringofFile( const AFile : string) : string');
3990: Procedure SaveStringtoFile( const AFile, AText : string)');
3991: Function LoadStringfromFile( const AFile : string) : string');
3992: function HexToColor(const AText: string): TColor;
3993: function UppercaseHTMLTags(const AText: string): string;
3994: function LowercaseHTMLTags(const AText: string): string;
3995: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
3996: function RelativePath(const ASrc, ADst: string): string;
3997: function GetToken(var Start: Integer; const SourceText: string): string;
3998: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
3999: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4000: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4001: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4002: // parses the beginning of an attribute: space + alpha character
4003: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4004: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4005: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4006: // parses all name=value attributes to the attributes TStringList
4007: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4008: // checks if a name="value" pair exists and returns any value
4009: function GetStrValue(const AText, AName, ADefault: string): string;
4010: // retrieves string value from a line like:
4011: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4012: // returns ADefault when not found
4013: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4014: // same for a color
4015: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4016: // same for an Integer
4017: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4018: // same for a float
4019: function GetBoolValue(const AText, AName: string): Boolean;
4020: // same for Boolean but without default
4021: function GetValue(const AText, AName: string): string;
4022: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4023: procedure SetValue(var AText: string; const AName, AValue: string);
4024: // sets a string value in a line
4025: procedure DeleteValue(var AText: string; const AName: string);
4026: // deletes a AName="value" pair from AText
4027: procedure GetNames(AText: string; AList: TStringList);
4028: // get a list of names from a string with name="value" pairs
4029: function GetHTMLColor(AColor: TColor): string;
4030: // converts a color value to the HTML hex value
4031: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4032: // finds a string backward case sensitive
4033: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
```

```
4034: // finds a string backward case insensitive
4035: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4036:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4037: // finds a text range, e.g. <TD>....</TD> case sensitive
4038: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4039:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4040: // finds a text range, e.g. <TD>....</td> case insensitive
4041: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4042:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4043: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4044: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4045:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4046: // finds a text range backward, e.g. <TD>....</td> case insensitive
4047: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4048:   var RangeEnd: Integer): Boolean;
4049: // finds a HTML or XML tag:  <....>
4050: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4051:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4052: // finds the innertext between opening and closing tags
4053: function Easter(NYear: Integer): TDateTime;
4054: // returns the easter date of a year.
4055: function GetWeekNumber(Today: TDateTime): string;
4056: //gets a datecode. Returns year and weeknumber in format: YYWW
4057: function ParseNumber(const S: string): Integer;
4058: // parse number returns the last position, starting from 1
4059: function ParseDate(const S: string): Integer;
4060: // parse a SQL style data string from positions 1,
4061: // starts and ends with #
4062:
4063: ****************************************unit JvJCLUtils;****************************************
4064:
4065: function VarIsInt(Value: Variant): Boolean;
4066:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4067: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4068: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4069: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4070: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4071: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4072: function GetWordOnPos(const S: string; const P: Integer): string;
4073: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4074: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4075: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4076: { GetWordOnPosEx working like GetWordOnPos function, but
4077:   also returns Word position in iBeg, iEnd variables }
4078: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4079: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4080: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4081: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4082: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4083: { GetEndPosCaret returns the caret position of the last char. For the position
4084:   after the last char of Text you must add 1 to the returned X value. }
4085: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4086: { GetEndPosCaret returns the caret position of the last char. For the position
4087:   after the last char of Text you must add 1 to the returned X value. }
4088: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4089: function SubStrBySeparator(const S:string;const Index:Integer;const Separator:string;StartIndex:Integer=
      1): string;
4090: function SubStrBySeparatorW(const S: WideString; const Index: Integer; const Separator: WideString;
      StartIndex: Integer = 1): WideString;
4091: { SubStrEnd same to previous function but Index numerated from the end of string }
4092: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4093: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4094: function SubWord(P: PChar; var P2: PChar): string;
4095: function CurrencyByWord(Value: Currency): string;
4096: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4097: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4098: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4099: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4100: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4101: { ReplaceString searches for all substrings, OldPattern,
4102:   in a string, S, and replaces them with NewPattern }
4103: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4104: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
      WideString;StartIndex:Integer=1):WideString;
4105: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4106: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4107: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4108: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4109:
4110: { Next 4 function for russian chars transliterating.
4111:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4112: procedure Dos2Win(var S: AnsiString);
4113: procedure Win2Dos(var S: AnsiString);
4114: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
```

```
4115: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4116: function Win2Koi(const S: AnsiString): AnsiString;
4117: { FillString fills the string Buffer with Count Chars }
4118: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4119: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4120: { MoveString copies Count Chars from Source to Dest }
4121: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE} overload;
4122: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4123:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4124: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4125: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4126: { MoveWideChar copies Count WideChars from Source to Dest }
4127: procedure MoveWideChar(const Source; var Dest; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4128: { FillNativeChar fills Buffer with Count NativeChars }
4129: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4130: { MoveWideChar copies Count WideChars from Source to Dest }
4131: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4132: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4133: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4134: { Spaces returns string consists on N space chars }
4135: function Spaces(const N: Integer): string;
4136: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4137: function AddSpaces(const S: string; const N: Integer): string;
4138: function SpacesW(const N: Integer): WideString;
4139: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4140: { function LastDateRUS for russian users only }
4141: { returns date relative to current date: 'âåà äíÿ íàçàä' }
4142: function LastDateRUS(const Dat: TDateTime): string;
4143: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4144: function CurrencyToStr(const Cur: Currency): string;
4145: { HasChar returns True, if Char, Ch, contains in string, S }
4146: function HasChar(const Ch: Char; const S: string): Boolean;
4147: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4148: function HasAnyChar(const Chars: string; const S: string): Boolean;
4149: {$IFNDEF COMPILER12_UP}
4150: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4151: {$ENDIF ~COMPILER12_UP}
4152: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
      SUPPORTS_INLINE}
4153: function CountOfChar(const Ch: Char; const S: string): Integer;
4154: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4155: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4156: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4157: function StrPosW(S, SubStr: PWideChar): PWideChar;
4158: function StrLenW(S: PWideChar): Integer;
4159: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4160: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4161: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4162: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )');
4163: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )');
4164:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat');
4165: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat');
4166: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)');
4167: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4168: Procedure GrayscaleBitmap( Bitmap : TBitmap)');
4169: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream');
4170: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)');
4171: Function ScreenPixelFormat : TPixelFormat');
4172: Function ScreenColorCount : Integer');
4173: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)');
4174: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint');
4175: //  SIRegister_TJvGradient(CL);
4176:
4177: {************************************ files routines}
4178: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4179: const
4180:   {$IFDEF MSWINDOWS}
4181:   DefaultCaseSensitivity = False;
4182:   {$ENDIF MSWINDOWS}
4183:   {$IFDEF UNIX}
4184:   DefaultCaseSensitivity = True;
4185:   {$ENDIF UNIX}
4186: { GenTempFileName returns temporary file name on
4187:   drive, there FileName is placed }
4188: function GenTempFileName(FileName: string): string;
4189: { GenTempFileNameExt same to previous function, but
4190:   returning filename has given extension, FileExt }
4191: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4192: { ClearDir clears folder Dir }
4193: function ClearDir(const Dir: string): Boolean;
```

```
4194: { DeleteDir clears and than delete folder Dir }
4195: function DeleteDir(const Dir: string): Boolean;
4196: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4197: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4198: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4199:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4200: function FileEquMasks(FileName, Masks: TFileName;
4201:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4202: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4203: {$IFDEF MSWINDOWS}
4204: { LZFileExpand expand file, FileSource,
4205:   into FileDest. Given file must be compressed, using MS Compress program }
4206: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4207: {$ENDIF MSWINDOWS}
4208: { FileGetInfo fills SearchRec record for specified file attributes}
4209: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4210: { HasSubFolder returns True, if folder APath contains other folders }
4211: function HasSubFolder(APath: TFileName): Boolean;
4212: { IsEmptyFolder returns True, if there are no files or
4213:   folders in given folder, APath}
4214: function IsEmptyFolder(APath: TFileName): Boolean;
4215: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4216: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4217: { AddPath returns FileName with Path, if FileName not contain any path }
4218: function AddPath(const FileName, Path: TFileName): TFileName;
4219: function AddPaths(const PathList, Path: string): string;
4220: function ParentPath(const Path: TFileName): TFileName;
4221: function FindInPath(const FileName, PathList: string): TFileName;
4222: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4223: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4224: { HasParam returns True, if program running with specified parameter, Param }
4225: function HasParam(const Param: string): Boolean;
4226: function HasSwitch(const Param: string): Boolean;
4227: function Switch(const Param: string): string;
4228: { ExePath returns ExtractFilePath(ParamStr(0)) }
4229: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4230: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4231: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4232: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4233: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4234: {**** Graphic routines }
4235: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4236: function IsTTFontSelected(const DC: HDC): Boolean;
4237: function KeyPressed(VK: Integer): Boolean;
4238: Function isKeypressed: boolean;');  //true if key on memo2 (shell output) is pressed
4239:
4240: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4241: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4242: {**** Color routines }
4243: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4244: function RGBToBGR(Value: Cardinal): Cardinal;
4245: //function ColorToPrettyName(Value: TColor): string;
4246: //function PrettyNameToColor(const Value: string): TColor;
4247: {**** other routines }
4248: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4249: function IntPower(Base, Exponent: Integer): Integer;
4250: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4251: function StrToBool(const S: string): Boolean;
4252: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4253: function VarToInt(V: Variant): Integer;
4254: function VarToFloat(V: Variant): Double;
4255: { following functions are not documented because they not work properly sometimes, so do not use them }
4256: // (rom) ReplaceStrings1, GetSubStr removed
4257: function GetLongFileName(const FileName: string): string;
4258: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4259: function GetParameter: string;
4260: function GetComputerID: string;
4261: function GetComputerName: string;
4262: {**** string routines }
4263: { ReplaceAllStrings searches for all substrings, Words,
4264:   in a string, S, and replaces them with Frases with the same Index. }
4265: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4266: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4267:   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 }
4268: function ReplaceStrings(const S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer):
      string;
4269: { CountOfLines calculates the lines count in a string, S,
4270:   each line must be separated from another with CrLf sequence }
4271: function CountOfLines(const S: string): Integer;
4272: { DeleteLines deletes all lines from strings which in the words,  words.
4273:   The word of will be deleted from strings. }
4274: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4275: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4276:   Lines contained only spaces also deletes. }
4277: procedure DeleteEmptyLines(Ss: TStrings);
```

```
4278: { SQLAddWhere addes or modifies existing where-statement, where,
4279:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4280:   it must be started on the begining of any line }
4281: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4282: {**** files routines - }
4283: {$IFDEF MSWINDOWS}
4284: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4285:   Resource can be compressed using MS Compress program}
4286: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4287: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
     Boolean;
4288: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4289: {$ENDIF MSWINDOWS}
4290: { IniReadSection read section, Section, from ini-file,
4291:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4292:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4293: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4294: { LoadTextFile load text file, FileName, into string }
4295: function LoadTextFile(const FileName: TFileName): string;
4296: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4297: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4298: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4299: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4300: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4301: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4302: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4303: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4304: { RATextCalcHeight calculate needed height for
4305:   correct output, using RATextOut or RATextOutEx functions }
4306: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4307: { Cinema draws some visual effect }
4308: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4309: { Roughed fills rect with special 3D pattern }
4310: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4311: { 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 }
4312: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4313: { TextWidth calculate text with for writing using standard desktop font }
4314: function TextWidth(const AStr: string): Integer;
4315: { TextHeight calculate text height for writing using standard desktop font }
4316: function TextHeight(const AStr: string): Integer;
4317: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4318: procedure Error(const Msg: string);
4319: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4320:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4321: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4322: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4323:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4324: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4325:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4326: function ItemHtPlain(const Text: string): string;
4327: { ClearList - clears list of TObject }
4328: procedure ClearList(List: TList);
4329: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
4330: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4331: { RTTI support }
4332: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4333: function GetPropStr(Obj: TObject; const PropName: string): string;
4334: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4335: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4336: procedure PrepareIniSection(Ss: TStrings);
4337: { following functions are not documented because they are don't work properly, so don't use them }
4338: // (rom) from JvBandWindows to make it obsolete
4339: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4340: // (rom) from JvBandUtils to make it obsolete
4341: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4342: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4343: function CreateIconFromClipboard: TIcon;
4344: { begin JvIconClipboardUtils } { Icon clipboard routines }
4345: function CF_ICON: Word;
4346: procedure AssignClipboardIcon(Icon: TIcon);
4347: { Real-size icons support routines (32-bit only) }
4348: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4349: function CreateRealSizeIcon(Icon: TIcon): HICON;
4350: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4351: {end JvIconClipboardUtils }
4352: function CreateScreenCompatibleDC: HDC;
4353: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF}
4354: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4355: { begin JvRLE } // (rom) changed API for inclusion in JCL
4356: procedure RleCompressTo(InStream, OutStream: TStream);
4357: procedure RleDecompressTo(InStream, OutStream: TStream);
4358: procedure RleCompress(Stream: TStream);
4359: procedure RleDecompress(Stream: TStream);
4360: { end JvRLE } { begin JvDateUtil }
```

```
4361: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4362: function IsLeapYear(AYear: Integer): Boolean;
4363: function DaysInAMonth(const AYear, AMonth: Word): Word;
4364: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4365: function FirstDayOfPrevMonth: TDateTime;
4366: function LastDayOfPrevMonth: TDateTime;
4367: function FirstDayOfNextMonth: TDateTime;
4368: function ExtractDay(ADate: TDateTime): Word;
4369: function ExtractMonth(ADate: TDateTime): Word;
4370: function ExtractYear(ADate: TDateTime): Word;
4371: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4372: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4373: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4374: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4375: function ValidDate(ADate: TDateTime): Boolean;
4376: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4377: function MonthsBetween(Date1, Date2: TDateTime): Double;
4378: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4379: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4380: function DaysBetween(Date1, Date2: TDateTime): Longint;
4381: { The same as previous but if Date2 < Date1 result = 0 }
4382: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4383: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4384: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4385: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4386: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4387: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4388: { String to date conversions }
4389: function GetDateOrder(const DateFormat: string): TDateOrder;
4390: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4391: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4392: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4393: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4394: //function DefDateFormat(AFourDigitYear: Boolean): string;
4395: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4396: function FormatLongDate(Value: TDateTime): string;
4397: function FormatLongDateTime(Value: TDateTime): string;
4398: { end JvDateUtil }
4399: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4400: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4401: { begin JvStrUtils } { ** Common string handling routines ** }
4402: {$IFDEF UNIX}
4403: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4404:   const ToCode, FromCode: AnsiString): Boolean;
4405: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4406: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4407: function OemStrToAnsi(const S: AnsiString): AnsiString;
4408: function AnsiStrToOem(const S: AnsiString): AnsiString;
4409: {$ENDIF UNIX}
4410: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4411: { StrToOem translates a string from the Windows character set into the OEM character set. }
4412: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4413: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4414: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4415: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4416: function ReplaceStr(const S, Srch, Replace: string): string;
4417: { Returns string with every occurrence of Srch string replaced with Replace string. }
4418: function DelSpace(const S: string): string;
4419: { DelSpace return a string with all white spaces removed. }
4420: function DelChars(const S: string; Chr: Char): string;
4421: { DelChars return a string with all Chr characters removed. }
4422: function DelBSpace(const S: string): string;
4423: { DelBSpace trims leading spaces from the given string. }
4424: function DelESpace(const S: string): string;
4425: { DelESpace trims trailing spaces from the given string. }
4426: function DelRSpace(const S: string): string;
4427: { DelRSpace trims leading and trailing spaces from the given string. }
4428: function DelSpace1(const S: string): string;
4429: { DelSpace1 return a string with all non-single white spaces removed. }
4430: function Tab2Space(const S: string; Numb: Byte): string;
4431: { Tab2Space converts any tabulation character in the given string to the
4432:   Numb spaces characters. }
4433: function NPos(const C: string; S: string; N: Integer): Integer;
4434: { NPos searches for a N-th position of substring C in a given string. }
4435: function MakeStr(C: Char; N: Integer): string; overload;
4436: {$IFNDEF COMPILER12_UP}
4437: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4438: {$ENDIF !COMPILER12_UP}
4439: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4440: { MakeStr return a string of length N filled with character C. }
4441: function AddChar(C: Char; const S: string; N: Integer): string;
4442: { AddChar return a string left-padded to length N with characters C. }
4443: function AddCharR(C: Char; const S: string; N: Integer): string;
4444: { AddCharR return a string right-padded to length N with characters C. }
4445: function LeftStr(const S: string; N: Integer): string;
4446: { LeftStr return a string right-padded to length N with blanks. }
```

```
4447: function RightStr(const S: string; N: Integer): string;
4448: { RightStr return a string left-padded to length N with blanks. }
4449: function CenterStr(const S: string; Len: Integer): string;
4450: { CenterStr centers the characters in the string based upon the Len specified. }
4451: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4452: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4453:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4454: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4455: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4456: function Copy2Symb(const S: string; Symb: Char): string;
4457: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4458: function Copy2SymbDel(var S: string; Symb: Char): string;
4459: { Copy2SymbDel returns a substring of a string S from begining to first
4460:   character Symb and removes this substring from S. }
4461: function Copy2Space(const S: string): string;
4462: { Copy2Symb returns a substring of a string S from begining to first white space. }
4463: function Copy2SpaceDel(var S: string): string;
4464: { Copy2SpaceDel returns a substring of a string S from begining to first
4465:   white space and removes this substring from S. }
4466: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4467: { Returns string, with the first letter of each word in uppercase,
4468:   all other letters in lowercase. Words are delimited by WordDelims. }
4469: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4470: { WordCount given a set of word delimiters, returns number of words in S. }
4471: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4472: { Given a set of word delimiters, returns start position of N'th word in S. }
4473: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4474: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4475: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4476: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4477:   delimiters, return the N'th word in S. }
4478: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4479: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4480:   that started from position Pos. }
4481: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4482: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4483: function QuotedString(const S: string; Quote: Char): string;
4484: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4485: function ExtractQuotedString(const S: string; Quote: Char): string;
4486: { ExtractQuotedString removes the Quote characters from the beginning and
4487:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4488: function FindPart(const HelpWilds, InputStr: string): Integer;
4489: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4490: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4491: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4492: function XorString(const Key, Src: ShortString): ShortString;
4493: function XorEncode(const Key, Source: string): string;
4494: function XorDecode(const Key, Source: string): string;
4495: { ** Command line routines ** }
4496: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4497: { ** Numeric string handling routines ** }
4498: function Numb2USA(const S: string): string;
4499: { Numb2USA converts numeric string S to USA-format. }
4500: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4501: { Dec2Hex converts the given value to a hexadecimal string representation
4502:   with the minimum number of digits (A) specified. }
4503: function Hex2Dec(const S: string): Longint;
4504: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4505: function Dec2Numb(N: Int64; A, B: Byte): string;
4506: { Dec2Numb converts the given value to a string representation with the
4507:   base equal to B and with the minimum number of digits (A) specified. }
4508: function Numb2Dec(S: string; B: Byte): Int64;
4509: { Numb2Dec converts the given B-based numeric string to the corresponding
4510:   integer value. }
4511: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4512: { IntToBin converts the given value to a binary string representation
4513:   with the minimum number of digits specified. }
4514: function IntToRoman(Value: Longint): string;
4515: { IntToRoman converts the given value to a roman numeric string representation. }
4516: function RomanToInt(const S: string): Longint;
4517: { RomanToInt converts the given string to an integer value. If the string
4518:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4519: function FindNotBlankCharPos(const S: string): Integer;
4520: function FindNotBlankCharPosW(const S: WideString): Integer;
4521: function AnsiChangeCase(const S: string): string;
4522: function WideChangeCase(const S: string): string;
4523: function StartsText(const SubStr, S: string): Boolean;
4524: function EndsText(const SubStr, S: string): Boolean;
4525: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4526: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4527: {end JvStrUtils}
4528: {$IFDEF UNIX}
4529: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4530: {$ENDIF UNIX}
4531: { begin JvFileUtil }
4532: function FileDateTime(const FileName: string): TDateTime;
```

```
4533: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4534: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4535: function NormalDir(const DirName: string): string;
4536: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4537: function ValidFileName(const FileName: string): Boolean;
4538: {$IFDEF MSWINDOWS}
4539: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4540: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4541: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4542: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4543: {$ENDIF MSWINDOWS}
4544: function GetWindowsDir: string;
4545: function GetSystemDir: string;
4546: function ShortToLongFileName(const ShortName: string): string;
4547: function LongToShortFileName(const LongName: string): string;
4548: function ShortToLongPath(const ShortName: string): string;
4549: function LongToShortPath(const LongName: string): string;
4550: {$IFDEF MSWINDOWS}
4551: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4552: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4553: {$ENDIF MSWINDOWS}
4554: { end JvFileUtil }
4555: // Works like PtInRect but includes all edges in comparision
4556: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4557: // Works like PtInRect but excludes all edges from comparision
4558: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4559: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4560: function IsFourDigitYear: Boolean;
4561: { moved from JvJVCLUTils }
4562: //Open an object with the shell (url or something like that)
4563: function OpenObject(const Value: string): Boolean; overload;
4564: function OpenObject(Value: PChar): Boolean; overload;
4565: {$IFDEF MSWINDOWS}
4566: //Raise the last Exception
4567: procedure RaiseLastWin32; overload;
4568: procedure RaiseLastWin32(const Text: string); overload;
4569: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
4570: function GetFileVersion(const AFileName: string): Cardinal;
4571: {$EXTERNALSYM GetFileVersion}
4572: //Get version of Shell.dll
4573: function GetShellVersion: Cardinal;
4574: {$EXTERNALSYM GetShellVersion}
4575: // CD functions  on HW
4576: procedure OpenCdDrive;
4577: procedure CloseCdDrive;
4578: // returns True if Drive is accessible
4579: function DiskInDrive(Drive: Char): Boolean;
4580: {$ENDIF MSWINDOWS}
4581: //Same as linux function ;)
4582: procedure PError(const Text: string);
4583: // execute a program without waiting
4584: procedure Exec(const FileName, Parameters, Directory: string);
4585: // execute a program and wait for it to finish
4586: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
4587: // returns True if this is the first instance of the program that is running
4588: function FirstInstance(const ATitle: string): Boolean;
4589: // restores a window based on it's classname and Caption. Either can be left empty
4590: // to widen the search
4591: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
4592: // manipulate the traybar and start button
4593: procedure HideTraybar;
4594: procedure ShowTraybar;
4595: procedure ShowStartButton(Visible: Boolean = True);
4596: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
4597: procedure MonitorOn;
4598: procedure MonitorOff;
4599: procedure LowPower;
4600: // send a key to the window named AppName
4601: function SendKey(const AppName: string; Key: Char): Boolean;
4602: {$IFDEF MSWINDOWS}
4603: // returns a list of all win currently visible, the Objects property is filled with their window handle
4604: procedure GetVisibleWindows(List: TStrings);
4605: // associates an extension to a specific program
4606: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
4607: procedure AddToRecentDocs(const FileName: string);
4608: function GetRecentDocs: TStringList;
4609: {$ENDIF MSWINDOWS}
4610: function CharIsMoney(const Ch: Char): Boolean;
4611: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
4612: function IntToExtended(I: Integer): Extended;
4613: { GetChangedText works out the new text given the current cursor pos & the key pressed
4614:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
4615: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
4616: function MakeYear4Digit(Year, Pivot: Integer): Integer;
```

```
4617: //function StrIsInteger(const S: string): Boolean;
4618: function StrIsFloatMoney(const Ps: string): Boolean;
4619: function StrIsDateTime(const Ps: string): Boolean;
4620: function PreformatDateString(Ps: string): string;
4621: function BooleanToInteger(const B: Boolean): Integer;
4622: function StringToBoolean(const Ps: string): Boolean;
4623: function SafeStrToDateTime(const Ps: string): TDateTime;
4624: function SafeStrToDate(const Ps: string): TDateTime;
4625: function SafeStrToTime(const Ps: string): TDateTime;
4626: function StrDelete(const psSub, psMain: string): string;
4627:   { returns the fractional value of pcValue}
4628: function TimeOnly(pcValue: TDateTime): TTime;
4629: { returns the integral value of pcValue }
4630: function DateOnly(pcValue: TDateTime): TDate;
4631: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
4632: const { TDateTime value used to signify Null value}
4633:   NullEquivalentDate: TDateTime = 0.0;
4634: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
4635: // Replacement for Win32Check to avoid platform specific warnings in D6
4636: function OSCheck(RetVal: Boolean): Boolean;
4637: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
4638:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
4639:   not be forced to use FileCtrl unnecessarily }
4640: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
4641: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
4642: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
4643: function MinimizeString(const S: string; const MaxLen: Integer): string;
4644: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
     SW_SHOWDEFAULT);
4645: { GetDLLVersion loads DLLName, gets a pointer to the DLLVersion function and calls it, returning the major
     and minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't
     be found.}
4646: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
4647: {$ENDIF MSWINDOWS}
4648: procedure ResourceNotFound(ResID: PChar);
4649: function EmptyRect: TRect;
4650: function RectWidth(R: TRect): Integer;
4651: function RectHeight(R: TRect): Integer;
4652: function CompareRect(const R1, R2: TRect): Boolean;
4653: procedure RectNormalize(var R: TRect);
4654: function RectIsSquare(const R: TRect): Boolean;
4655: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
4656: //If AMaxSize = -1 ,then auto calc Square's max size
4657: {$IFDEF MSWINDOWS}
4658: procedure FreeUnusedOle;
4659: function GetWindowsVersion: string;
4660: function LoadDLL(const LibName: string): THandle;
4661: function RegisterServer(const ModuleName: string): Boolean;
4662: function UnregisterServer(const ModuleName: string): Boolean;
4663: {$ENDIF MSWINDOWS}
4664: { String routines }
4665: function GetEnvVar(const VarName: string): string;
4666: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
4667: function StringToPChar(var S: string): PChar;
4668: function StrPAlloc(const S: string): PChar;
4669: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
4670: function DropT(const S: string): string;
4671: { Memory routines }
4672: function AllocMemo(Size: Longint): Pointer;
4673: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
4674: procedure FreeMemo(var fpBlock: Pointer);
4675: function GetMemoSize(fpBlock: Pointer): Longint;
4676: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
4677: { Manipulate huge pointers routines }
4678: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
4679: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
4680: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
4681: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
4682: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
4683: function WindowClassName(Wnd: THandle): string;
4684: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
4685: procedure ActivateWindow(Wnd: THandle);
4686: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
4687: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
4688: { SetWindowTop put window to top without recreating window }
4689: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
4690: procedure CenterWindow(Wnd: THandle);
4691: function MakeVariant(const Values: array of Variant): Variant;
4692: { Convert dialog units to pixels and backwards }
4693: {$IFDEF MSWINDOWS}
4694: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
4695: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
4696: function PixelsToDialogUnitsX(PixUnits: Word): Word;
4697: function PixelsToDialogUnitsY(PixUnits: Word): Word;
4698: {$ENDIF MSWINDOWS}
4699: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
```

```
{$IFDEF BCB}
function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
{$ELSE}
function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
{$ENDIF BCB}
{$IFDEF MSWINDOWS}
{ BrowseForFolderNative displays Browse For Folder dialog }
function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
{$ENDIF MSWINDOWS}
procedure AntiAlias(Clip: TBitmap);
procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
  ABitmap: TBitmap; const SourceRect: TRect);
function IsTrueType(const FontName: string): Boolean;
// Removes all non-numeric characters from AValue and returns the resulting string
function TextToValText(const AValue: string): string;
Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean');
Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)');
Function ReplaceRegExpr(const ARegExpr,AInputStr, AReplaceStr: RegExprString; AUseSubstitution : boolean)
: RegExprString');
Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString');
Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :

**********************************************************unit uPSI_JvTFUtils;
 Function JExtractYear( ADate : TDateTime) : Word');
 Function JExtractMonth( ADate : TDateTime) : Word');
 Function JExtractDay( ADate : TDateTime) : Word');
 Function ExtractHours( ATime : TDateTime) : Word');
 Function ExtractMins( ATime : TDateTime) : Word');
 Function ExtractSecs( ATime : TDateTime) : Word');
 Function ExtractMSecs( ATime : TDateTime) : Word');
 Function FirstOfMonth( ADate : TDateTime) : TDateTime');
 Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word');
 Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word');
 Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)');
 Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)');
 Procedure IncDays( var ADate : TDateTime; N : Integer)');
 Procedure IncWeeks( var ADate : TDateTime; N : Integer)');
 Procedure IncMonths( var ADate : TDateTime; N : Integer)');
 Procedure IncYears( var ADate : TDateTime; N : Integer)');
 Function EndOfMonth( ADate : TDateTime) : TDateTime');
 Function IsFirstOfMonth( ADate : TDateTime) : Boolean');
 Function IsEndOfMonth( ADate : TDateTime) : Boolean');
 Procedure EnsureMonth( Month : Word)');
 Procedure EnsureDOW( DOW : Word)');
 Function EqualDates( D1, D2 : TDateTime) : Boolean');
 Function Lesser( N1, N2 : Integer) : Integer');
 Function Greater( N1, N2 : Integer) : Integer');
 Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer');
 Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer');
 Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer');
 Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer');
 Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek');
 Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek');
 Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop : Integer; var TextBounds : TRect;
AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)');
 Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)');
 Function JRectWidth( ARect : TRect) : Integer');
 Function JRectHeight( ARect : TRect) : Integer');
 Function JEmptyRect : TRect');
 Function IsClassByName( Obj : TObject; ClassName : string) : Boolean');


*******************************Integer Huge Cardinal Utils**************************
Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64');
Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32');
Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64');
Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32');
Function BitCount_8( Value : byte) : integer');
Function BitCount_16( Value : uint16) : integer');
Function BitCount_32( Value : uint32) : integer');
Function BitCount_64( Value : uint64) : integer');
Function CountSetBits_64( Value : uint64) : integer'); TPrimalityTestNoticeProc',
 Procedure ( CountPrimalityTests : integer)');
 Function gcd( a, b : THugeCardinal) : THugeCardinal');
 Function lcm( a, b : THugeCardinal) : THugeCardinal');
 Function isCoPrime( a, b : THugeCardinal) : boolean');
 Function isProbablyPrime(p: THugeCardinal;OnProgress: TProgress; var wasAborted: boolean): boolean');
 Function hasSmallFactor( p : THugeCardinal) : boolean');
 //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool;var Prime: THugeCardinal; var
NumbersTested: integer) : boolean');
 Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal');
```

```
4781:  AddConstantN('StandardExponent','LongInt').SetInt( 65537);
4782:  //Procedure Compute_RSA_Fundamentals_2Factors( RequiredBitLengthOfN : integer;Fixed_e:uint64; var N, e, d,
       Totient : TProgress;
      OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:integer;Pool1:TMemoryStreamPool;var Numbers
4783:  Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
4784:
4785: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
4786: begin
4787:   AddTypeS('TXRTLInteger', 'array of Integer');
4788:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException');
4789:   CL.AddClassN(FindClass('TOBJECT'),'EXRTLExtendInvalidArgument');
4790:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero');
4791:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument');
4792:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix');
4793:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit');
4794:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument');
4795:   AddConstantN('BitsPerByte','LongInt').SetInt( 8);
4796:   BitsPerDigit','LongInt').SetInt( 32);
4797:   SignBitMask','LongWord').SetUInt( $80000000);
4798:  Function XRTLAdjustBits( const ABits : Integer) : Integer');
4799:  Function XRTLLength( const AInteger : TXRTLInteger) : Integer');
4800:  Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer');
4801:  Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)');
4802:  Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)');
4803:  Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)');
4804:  Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer');
4805:  Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean');
4806:  Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Integer;var
      AResult:TXRTLInteger):Integer;
4807:  Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
4808:  Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var
      AResult:TXRTLInteger):Integer');
4809:  Function XRTLSignStrip( const AInteger: TXRTLInteger; var AResult:TXRTLInteger; const
      AMinDataBits:Integer) : Integer');
4810:  Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)');
4811:  Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)');
4812:  Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)');
4813:  Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)');
4814:  Function XRTLSign( const AInteger : TXRTLInteger) : Integer');
4815:  Procedure XRTLZero( var AInteger : TXRTLInteger)');
4816:  Procedure XRTLOne( var AInteger : TXRTLInteger)');
4817:  Procedure XRTLMOne( var AInteger : TXRTLInteger)');
4818:  Procedure XRTLTwo( var AInteger : TXRTLInteger)');
4819:  Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer');
4820:  Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer');
4821:  Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)');
4822:  Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;');
4823:  Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;');
4824:  Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;');
4825:  Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
4826:  Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;');
4827:  Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;');
4828:  Function XRTLUMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer');
4829:  Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer');
4830:  Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer');
4831:  Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)');
4832:  Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)');
4833:  Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)');
4834:  Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)');
4835:  Procedure XRTLRootApprox( const AInteger1, AInteger2 : TXRTLInteger; var ALowApproxResult,
      AHighApproxResult : TXRTLInteger)');
4836:  Procedure XRTLURootApprox( const AInteger1, AInteger2 : TXRTLInteger; var ALowApproxResult,
      AHighApproxResult : TXRTLInteger)');
4837:  Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)');
4838:  Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)');
4839:  Procedure XRTLSLBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)');
4840:  Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)');
4841:  Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)');
4842:  Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)');
4843:  Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)');
4844:  Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)');
4845:  Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)');
4846:  Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)');
4847:  Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)');
4848:  Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)');
4849:  Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)');
4850:  Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)');
4851:  Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string');
4852:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string');
4853:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string');
4854:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)');
4855:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)');
4856:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)');
4857:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);');
4858:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);');
4859:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);');
```

```
4860:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)');
4861:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)');
4862:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer');
4863:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)');
4864:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);');
4865:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);');
4866:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);');
4867:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);');
4868:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)');
4869:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)');
4870:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)');
4871:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)');
4872: end;
4873:
4874: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
4875: begin
4876:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean');
4877:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)');
4878:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
4879:  Procedure JvXPAdjustBoundRect( const BorderWidth : Byte; const ShowBoundLines : Boolean; const BoundLines
      : TJvXPBoundLines; var Rect : TRect)');
4880:  Procedure JvXPDrawBoundLines( const ACanvas : TCanvas; const BoundLines : TJvXPBoundLines; const AColor :
      TColor; const Rect : TRect)');
4881:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)');
4882:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)');
4883:  Procedure JvXPFrame3D( const ACanvas : TCanvas; const Rect : TRect; const TopColor, BottomColor : TColor;
      const Swapped : Boolean)');
4884:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)');
4885:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)');
4886:  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)');
4887: end;
4888:
4889:
4890: ********unit uPSI_StUtils; from Systools4************************************************************
4891: Function SignL( L : LongInt) : Integer');
4892: Function SignF( F : Extended) : Integer');
4893: Function MinWord( A, B : Word) : Word');
4894: Function MidWord( W1, W2, W3 : Word) : Word');
4895: Function MaxWord( A, B : Word) : Word');
4896: Function MinLong( A, B : LongInt) : LongInt');
4897: Function MidLong( L1, L2, L3 : LongInt) : LongInt');
4898: Function MaxLong( A, B : LongInt) : LongInt');
4899: Function MinFloat( F1, F2 : Extended) : Extended');
4900: Function MidFloat( F1, F2, F3 : Extended) : Extended');
4901: Function MaxFloat( F1, F2 : Extended) : Extended');
4902: Function MakeInteger16( H, L : Byte) : SmallInt');
4903: Function MakeWordS( H, L : Byte) : Word');
4904: Function SwapNibble( B : Byte) : Byte');
4905: Function SwapWord( L : LongInt) : LongInt');
4906: Procedure SetFlag( var Flags : Word; FlagMask : Word)');
4907: Procedure ClearFlag( var Flags : Word; FlagMask : Word)');
4908: Function FlagIsSet( Flags, FlagMask : Word) : Boolean');
4909: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)');
4910: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)');
4911: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean');
4912: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)');
4913: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)');
4914: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean');
4915: Procedure ExchangeBytes( var I, J : Byte)');
4916: Procedure ExchangeWords( var I, J : Word)');
4917: Procedure ExchangeLongInts( var I, J : LongInt)');
4918: Procedure ExchangeStructs( var I, J, Size : Cardinal)');
4919: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)');
4920: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)');
4921: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer');
4922: //*****************uPSI_StFIN;************************************************************
4923: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended');
4924: Function AccruedInterestPeriodic( Issue, Settlement, Maturity : TStDate; Rate, Par : Extended; Frequency :
      TStFrequency; Basis : TStBasis) : Extended');
4925: Function BondDuration( Settlement, Maturity : TStDate; Rate, Yield : Extended; Frequency : TStFrequency;
      Basis : TStBasis) : Extended');
4926: Function BondPrice( Settlement, Maturity : TStDate; Rate, Yield, Redemption : Extended; Frequency :
      TStFrequency; Basis : TStBasis) : Extended');
4927: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended');
4928: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended');
4929: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt');
4930: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended');
4931: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
4932: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended');
4933: Function DollarToDecimalText( DecDollar : Extended) : string');
```

```
4934: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended');
4935: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string');
4936: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended');
4937: Function FutureValueS(Rate:Extended;NPeriods:Integer;Pmt,
      PV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Extended');
4938: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended');
4939: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended');
4940: Function InterestRateS(NPeriods: Integer; Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime;
      Guess:Extended) : Extended');
4941: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended');
4942: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended');
4943: Function IsCardValid( const S : string) : Boolean');
4944: Function ModifiedDuration( Settlement, Maturity : TStDate; Rate, Yield : Extended; Frequency :
      TStFrequency; Basis : TStBasis) : Extended');
4945: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended');
4946: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended');
4947: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended');
4948: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended');
4949: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended');
4950: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
4951: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
4952: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended');
4953: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
4954: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended');
4955: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
4956: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended');
4957: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended');
4958: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended');
4959: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended');
4960: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended');
4961: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
4962: Function YieldPeriodic( Settlement, Maturity : TStDate; Rate, Price, Redemption : Extended; Frequency :
      TStFrequency; Basis : TStBasis) : Extended');
4963: Function YieldMaturity(Issue,Settlement,Maturity: TStDate; Rate,Price:Extended; Basis:TStBasis):
      Extended');
4964:
4965: //*****************************************unit uPSI_StAstroP;
4966: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)');
4967: //*****unit unit uPSI_StStat; Statistic Package of SysTools****************************************
4968: Function AveDev( const Data : array of Double) : Double');
4969: Function AveDev16( const Data, NData : Integer) : Double');
4970: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double');
4971: Function Correlation( const Data1, Data2 : array of Double) : Double');
4972: Function Correlation16( const Data1, Data2, NData : Integer) : Double');
4973: Function Covariance( const Data1, Data2 : array of Double) : Double');
4974: Function Covariance16( const Data1, Data2, NData : Integer) : Double');
4975: Function DevSq( const Data : array of Double) : Double');
4976: Function DevSq16( const Data, NData : Integer) : Double');
4977: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
4978:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)');
4979: Function GeometricMeanS( const Data : array of Double) : Double');
4980: Function GeometricMean16( const Data, NData : Integer) : Double');
4981: Function HarmonicMeanS( const Data : array of Double) : Double');
4982: Function HarmonicMean16( const Data, NData : Integer) : Double');
4983: Function Largest( const Data : array of Double; K : Integer) : Double');
4984: Function Largest16( const Data, NData : Integer; K : Integer) : Double');
4985: Function MedianS( const Data : array of Double) : Double');
4986: Function Median16( const Data, NData : Integer) : Double');
4987: Function Mode( const Data : array of Double) : Double');
4988: Function Mode16( const Data, NData : Integer) : Double');
4989: Function Percentile( const Data : array of Double; K : Double) : Double');
4990: Function Percentile16( const Data, NData : Integer; K : Double) : Double');
4991: Function PercentRank( const Data : array of Double; X : Double) : Double');
4992: Function PercentRank16( const Data, NData : Integer; X : Double) : Double');
4993: Function Permutations( Number, NumberChosen : Integer) : Extended');
4994: Function Combinations( Number, NumberChosen : Integer) : Extended');
4995: Function FactorialS( N : Integer) : Extended');
4996: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer');
4997: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer');
4998: Function Smallest( const Data : array of Double; K : Integer) : Double');
4999: Function Smallest16( const Data, NData : Integer; K : Integer) : Double');
5000: Function TrimMean( const Data : array of Double; Percent : Double) : Double');
5001: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double');
5002:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5003:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer; end');
5004: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Boolean);
5005: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Boolean);
5006: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double');
5007: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5008: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double');
5009: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double');
```

```
5010: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double');
5011: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double');
5012: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single');
5013: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single');
5014: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single');
5015: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer');
5016: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single');
5017: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single');
5018: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single');
5019: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single');
5020: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single');
5021: Function LogNormDist( X, Mean, StandardDev : Single) : Single');
5022: Function LogInv( Probability, Mean, StandardDev : Single) : Single');
5023: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single');
5024: Function NormInv( Probability, Mean, StandardDev : Single) : Single');
5025: Function NormSDist( Z : Single) : Single');
5026: Function NormSInv( Probability : Single) : Single');
5027: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single');
5028: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single');
5029: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single');
5030: Function Erfc( X : Single) : Single');
5031: Function GammaLn( X : Single) : Single');
5032: Function LargestSort( const Data : array of Double; K : Integer) : Double');
5033: Function SmallestSort( const Data : array of double; K : Integer) : Double');
5034:
5035: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5036:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt');
5037:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt');
5038:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo');
5039:  Function DefaultMergeName( MergeNum : Integer) : string');
5040:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)');
5041:
5042: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5043: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime');
5044:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec');
5045:  Function SunPos( UT : TStDateTimeRec) : TStPosRec');
5046:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec');
5047:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec');
5048:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType : TStTwilight) : TStRiseSetRec');
5049:  Function LunarPhase( UT : TStDateTimeRec) : Double');
5050:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec');
5051:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec');
5052:  Function FirstQuarter( D : TStDate) : TStLunarRecord');
5053:  Function FullMoon( D : TStDate) : TStLunarRecord');
5054:  Function LastQuarter( D : TStDate) : TStLunarRecord');
5055:  Function NewMoon( D : TStDate) : TStLunarRecord');
5056:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec');
5057:  Function NextFullMoon( D : TStDate) : TStDateTimeRec');
5058:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec');
5059:  Function NextNewMoon( D : TStDate) : TStDateTimeRec');
5060:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec');
5061:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec');
5062:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec');
5063:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec');
5064:  Function SiderealTime( UT : TStDateTimeRec) : Double');
5065:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec');
5066:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec');
5067:  Function SEaster( Y, Epoch : Integer) : TStDate');
5068:  Function DateTimeToAJD( D : TDateTime) : Double');
5069:  Function HoursMin( RA : Double) : ShortString');
5070:  Function DegsMin( DC : Double) : ShortString');
5071:  Function AJDToDateTime( D : Double) : TDateTime');
5072:
5073: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5074:  Function CurrentDate : TStDate');
5075:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean');
5076:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate');
5077:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)');
5078:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate');
5079:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate');
5080:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)');
5081:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate');
5082:  Function WeekOfYear( Julian : TStDate) : Byte');
5083:  Function AstJulianDate( Julian : TStDate) : Double');
5084:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate');
5085:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double');
5086:  Function StDayOfWeek( Julian : TStDate) : TStDayType');
5087:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType');
5088:  Function StIsLeapYear( Year : Integer) : Boolean');
5089:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer');
5090:  Function ResolveEpoch( Year, Epoch : Integer) : Integer');
5091:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean');
5092:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)');
5093:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime');
5094:  Function CurrentTime : TStTime');
5095:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)');
```

```
5096:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime');
5097:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime');
5098:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime');
5099:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime');
5100:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5101:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5102:  Function DateTimeToStDate( DT : TDateTime) : TStDate');
5103:  Function DateTimeToStTime( DT : TDateTime) : TStTime');
5104:  Function StDateToDateTime( D : TStDate) : TDateTime');
5105:  Function StTimeToDateTime( T : TStTime) : TDateTime');
5106:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate');
5107:  Function Convert4ByteDate( FourByteDate : TStDate) : Word');
5108:
5109: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5110: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double');
5111:  Function MonthToString( const Month : Integer) : string');
5112:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate');
5113:  Function DateStringToDMY( const Picture, S : string; Epoch : Integer; var D, M, Y : Integer):Boolean');
5114:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string');
5115:  Function DayOfWeekToString( const WeekDay : TStDayType) : string');
5116:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5117:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string');
5118:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string');
5119:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean');
5120:  Function TimeStringToStTime( const Picture, S : string) : TStTime');
5121:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string');
5122:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string');
5123:  Function DateStringIsBlank( const Picture, S : string) : Boolean');
5124:  Function InternationalDate( ForceCentury : Boolean) : string');
5125:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string');
5126:  Function InternationalTime( ShowSeconds : Boolean) : string');
5127:  Procedure ResetInternationalInfo');
5128:
5129: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5130:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean');
5131:  Function AnsiUpperCaseShort32( const S : string) : string');
5132:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer');
5133:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer');
5134:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint');
5135:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer, OutLen : LongInt) : Longint');
5136:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)');
5137:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)');
5138:  Function Upcase( C : AnsiChar) : AnsiChar');
5139:  Function LoCase( C : AnsiChar) : AnsiChar');
5140:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal');
5141:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer');
5142:  Function Search(const Buffer,BufLength: Cardinal; const Match,MatLength: Cardinal; var Pos: Cardinal) :
      Boolean');
5143:  Function SearchUC(const Buffer,BufLength: Cardinal; const Match,MatLength: Cardinal; var Pos: Cardinal):
      Boolean');
5144:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean');
5145:  Procedure RaiseContainerError( Code : longint)');
5146:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)');
5147:  Function ProductOverflow( A, B : LongInt) : Boolean');
5148:  Function StNewStr( S : string) : PShortString');
5149:  Procedure StDisposeStr( PS : PShortString)');
5150:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)');
5151:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)');
5152:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)');
5153:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)');
5154:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)');
5155:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)');
5156:
5157:
5158: //**********unit unit ; StMath Package of SysTools*****************************************
5159: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended');
5160: Function PowerS( Base, Exponent : Extended) : Extended');
5161: Function StInvCos( X : Double) : Double');
5162: Function StInvSin( Y : Double) : Double');
5163: Function StInvTan2( X, Y : Double) : Double');
5164: Function StTan( A : Double) : Double');
5165: Procedure DumpException;  //unit StExpEng;
5166: Function HexifyBlock( var Buffer, BufferSize : Integer) : string');
5167:
5168: //**********unit unit ; StCRC Package of SysTools*****************************************
5169: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt');
5170: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt');
5171: Function Adler32OfFile( FileName : AnsiString) : LongInt');
5172: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal');
5173: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal');
5174: Function Crc16OfFile( FileName : AnsiString) : Cardinal');
5175: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt');
5176: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt');
5177: Function Crc32OfFile( FileName : AnsiString) : LongInt');
5178: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal');
5179: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal');
```

```
5180: Function InternetSumOfFile( FileName : AnsiString) : Cardinal');
5181: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal');
5182: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal');
5183: Function Kermit16OfFile( FileName : AnsiString) : Cardinal');
5184:
5185: //**********unit unit ; StBCD Package of SysTools*****************************************
5186: Function AddBcd( const B1, B2 : TbcdS) : TbcdS');
5187: Function SubBcd( const B1, B2 : TbcdS) : TbcdS');
5188: Function MulBcd( const B1, B2 : TbcdS) : TbcdS');
5189: Function DivBcd( const B1, B2 : TbcdS) : TbcdS');
5190: Function ModBcd( const B1, B2 : TbcdS) : TbcdS');
5191: Function NegBcd( const B : TbcdS) : TbcdS');
5192: Function AbsBcd( const B : TbcdS) : TbcdS');
5193: Function FracBcd( const B : TbcdS) : TbcdS');
5194: Function IntBcd( const B : TbcdS) : TbcdS');
5195: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS');
5196: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS');
5197: Function ValBcd( const S : string) : TbcdS');
5198: Function LongBcd( L : LongInt) : TbcdS');
5199: Function ExtBcd( E : Extended) : TbcdS');
5200: Function ExpBcd( const B : TbcdS) : TbcdS');
5201: Function LnBcd( const B : TbcdS) : TbcdS');
5202: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS');
5203: Function PowBcd( const B, E : TbcdS) : TbcdS');
5204: Function SqrtBcd( const B : TbcdS) : TbcdS');
5205: Function CmpBcd( const B1, B2 : TbcdS) : Integer');
5206: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean');
5207: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean');
5208: Function IsIntBcd( const B : TbcdS) : Boolean');
5209: Function TruncBcd( const B : TbcdS) : LongInt');
5210: Function BcdExt( const B : TbcdS) : Extended');
5211: Function RoundBcd( const B : TbcdS) : LongInt');
5212: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string');
5213: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string');
5214: Function FormatBcd( const Format : string; const B : TbcdS) : string');
5215: Function StrGeneralBcd( const B : TbcdS) : string');
5216: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string');
5217: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)');
5218:
5219: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
5220: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)');
5221: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString');
5222: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType');
5223: Function StDeEscape( const EscStr : AnsiString) : Char');
5224: Function StDoEscape( Delim : Char) : AnsiString');
5225: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString');
5226: Function AnsiHashText( const S : string; Size : Integer) : Integer');
5227: Function AnsiHashStr( const S : string; Size : Integer) : Integer');
5228: Function AnsiELFHashText( const S : string; Size : Integer) : Integer');
5229: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer');
5230:
5231: //**********unit unit ; StNetCon Package of SysTools*****************************************
5232:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
5233:     Constructor Create( AOwner : TComponent)');
5234:     Function Connect : DWord');
5235:     Function Disconnect : DWord');
5236:     RegisterProperty('Password', 'String', iptrw);
5237:     Property('UserName', 'String', iptrw);
5238:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
5239:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
5240:     Property('LocalDevice', 'String', iptrw);
5241:     Property('ServerName', 'String', iptrw);
5242:     Property('ShareName', 'String', iptrw);
5243:     Property('OnConnect', 'TNotifyEvent', iptrw);
5244:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
5245:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
5246:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
5247:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
5248:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
5249:   end;
5250: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
5251: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
5252: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)');
5253: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)');
5254: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)');
5255: Function InitializeCriticalSectionAndSpinCount(var
      lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
5256: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
5257: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL');
5258: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)');
5259: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL');
5260: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL');
5261: Function SuspendThread( hThread : THandle) : DWORD');
5262: Function ResumeThread( hThread : THandle) : DWORD');
5263: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle');
5264: Function GetCurrentThread : THandle');
```

```
5265: Procedure ExitThread( dwExitCode : DWORD)');
5266: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL');
5267: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL');
5268: Procedure EndThread(ExitCode: Integer);');
5269: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD');
5270: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC');
5271: Procedure FreeProcInstance( Proc : FARPROC)');
5272: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)');
5273: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL');
5274: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);');
5275: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);');
5276: Procedure
     ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Pointer;AParam:Pointer;ASafeSection:boolean);
5277: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);');
5278: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Pointer;ASafeSection:boolean):TParallelJob;
5279: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;');
5280: Function CurrentParallelJobInfo : TParallelJobInfo');
5281: Function ObtainParallelJobInfo : TParallelJobInfo');
5282:
5283: ************************************************unit uPSI_JclMime;
5284:  Function MimeEncodeString( const S : AnsiString) : AnsiString');
5285:  Function MimeDecodeString( const S : AnsiString) : AnsiString');
5286:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)');
5287:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)');
5288:  Function MimeEncodedSize( const I : Cardinal) : Cardinal');
5289:  Function MimeDecodedSize( const I : Cardinal) : Cardinal');
5290:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)');
5291:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
5292:  Function MimeDecodePartial( var InputBuffer : string; const InputBytesCount : Cardinal; var OutputBuffer
     : string; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal');
5293:  Function MimeDecodePartialEnd(var OutputBuffer: string;const ByteBuffer: Cardinal;const
     ByteBufferSpace:Cardinal): Cardinal');
5294:
5295: ************************************************unit uPSI_JclPrint;
5296:  Procedure DirectPrint( const Printer, Data : string)');
5297:  Procedure SetPrinterPixelsPerInch');
5298:  Function GetPrinterResolution : TPoint');
5299:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer');
5300:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)');
5301:
5302:
5303: //************************************unit uPSI_ShLwApi;***********************************
5304:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar');
5305:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar');
5306:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer');
5307:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer');
5308:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer');
5309:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer');
5310:  Function StrDup( lpSrch : PChar) : PChar');
5311:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar');
5312:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar');
5313:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer');
5314:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL');
5315: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar');
5316:  Function StrPBrk( psz, pszSet : PChar) : PChar');
5317:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar');
5318:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar');
5319:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar');
5320:  Function StrSpn( psz, pszSet : PChar) : Integer');
5321:  Function StrStr( lpFirst, lpSrch : PChar) : PChar');
5322:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar');
5323:  Function StrToInt( lpSrch : PChar) : Integer');
5324:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL');
5325:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL');
5326:  Function ChrCmpI( w1, w2 : WORD) : BOOL');
5327:  Function ChrCmpIA( w1, w2 : WORD) : BOOL');
5328:  Function ChrCmpIW( w1, w2 : WORD) : BOOL');
5329:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL');
5330:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL');
5331:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar');
5332:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar');
5333:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL');
5334:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL');
5335:  SZ_CONTENTTYPE_HTMLA','String').SetString( 'text/html');
5336:  SZ_CONTENTTYPE_HTMLW','String').SetString( 'text/html');
5337:  SZ_CONTENTTYPE_HTML','string').SetString( SZ_CONTENTTYPE_HTMLA);
5338:  SZ_CONTENTTYPE_CDFA','String').SetString( 'application/x-cdf');
5339:  SZ_CONTENTTYPE_CDFW','String').SetString( 'application/x-cdf');
5340:  SZ_CONTENTTYPE_CDF','string').SetString( SZ_CONTENTTYPE_CDFA);
5341:  Function PathIsHTMLFile( pszPath : PChar) : BOOL');
5342:  STIF_DEFAULT','LongWord').SetUInt( $00000000);
5343:  STIF_SUPPORT_HEX','LongWord').SetUInt( $00000001);
5344:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer');
5345:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar');
5346:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar');
5347:  Function PathAddBackslash( pszPath : PChar) : PChar');
```

```
5348:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL');
5349:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL');
5350:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar');
5351:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL');
5352:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar');
5353:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL');
5354:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags : DWORD) : BOOL');
5355:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer');
5356:  Function PathFileExists( pszPath : PChar) : BOOL');
5357:  Function PathFindExtension( pszPath : PChar) : PChar');
5358:  Function PathFindFileName( pszPath : PChar) : PChar');
5359:  Function PathFindNextComponent( pszPath : PChar) : PChar');
5360:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL');
5361:  Function PathGetArgs( pszPath : PChar) : PChar');
5362:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar');
5363:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL');
5364:  Function PathGetCharType( ch : Char) : UINT');
5365:  GCT_INVALID','LongWord').SetUInt( $0000);
5366:  GCT_LFNCHAR','LongWord').SetUInt( $0001);
5367:  GCT_SHORTCHAR','LongWord').SetUInt( $0002);
5368:  GCT_WILD','LongWord').SetUInt( $0004);
5369:  GCT_SEPARATOR','LongWord').SetUInt( $0008);
5370:  Function PathGetDriveNumber( pszPath : PChar) : Integer');
5371:  Function PathIsDirectory( pszPath : PChar) : BOOL');
5372:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL');
5373:  Function PathIsFileSpec( pszPath : PChar) : BOOL');
5374:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL');
5375:  Function PathIsRelative( pszPath : PChar) : BOOL');
5376:  Function PathIsRoot( pszPath : PChar) : BOOL');
5377:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL');
5378:  Function PathIsUNC( pszPath : PChar) : BOOL');
5379:  Function PathIsNetworkPath( pszPath : PChar) : BOOL');
5380:  Function PathIsUNCServer( pszPath : PChar) : BOOL');
5381:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL');
5382:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL');
5383:  Function PathIsURL( pszPath : PChar) : BOOL');
5384:  Function PathMakePretty( pszPath : PChar) : BOOL');
5385:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL');
5386:  Function PathParseIconLocation( pszIconFile : PChar) : Integer');
5387:  Procedure PathQuoteSpaces( lpsz : PChar)');
5388:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
5389:  Procedure PathRemoveArgs( pszPath : PChar)');
5390:  Function PathRemoveBackslash( pszPath : PChar) : PChar');
5391:  Procedure PathRemoveBlanks( pszPath : PChar)');
5392:  Procedure PathRemoveExtension( pszPath : PChar)');
5393:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL');
5394:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL');
5395:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL');
5396:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)');
5397:  Function PathSkipRoot( pszPath : PChar) : PChar');
5398:  Procedure PathStripPath( pszPath : PChar)');
5399:  Function PathStripToRoot( pszPath : PChar) : BOOL');
5400:  Procedure PathUnquoteSpaces( lpsz : PChar)');
5401:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL');
5402:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL');
5403:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL');
5404:  Procedure PathUndecorate( pszPath : PChar)');
5405:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL');
5406:  URL_SCHEME_INVALID','LongInt').SetInt( - 1);
5407:  URL_SCHEME_UNKNOWN','LongInt').SetInt( 0);
5408:  URL_SCHEME_FTP','LongInt').SetInt( 1);
5409:  URL_SCHEME_HTTP','LongInt').SetInt( 2);
5410:  URL_SCHEME_GOPHER','LongInt').SetInt( 3);
5411:  URL_SCHEME_MAILTO','LongInt').SetInt( 4);
5412:  URL_SCHEME_NEWS','LongInt').SetInt( 5);
5413:  URL_SCHEME_NNTP','LongInt').SetInt( 6);
5414:  URL_SCHEME_TELNET','LongInt').SetInt( 7);
5415:  URL_SCHEME_WAIS','LongInt').SetInt( 8);
5416:  URL_SCHEME_FILE','LongInt').SetInt( 9);
5417:  URL_SCHEME_MK','LongInt').SetInt( 10);
5418:  URL_SCHEME_HTTPS','LongInt').SetInt( 11);
5419:  URL_SCHEME_SHELL','LongInt').SetInt( 12);
5420:  URL_SCHEME_SNEWS','LongInt').SetInt( 13);
5421:  URL_SCHEME_LOCAL','LongInt').SetInt( 14);
5422:  URL_SCHEME_JAVASCRIPT','LongInt').SetInt( 15);
5423:  URL_SCHEME_VBSCRIPT','LongInt').SetInt( 16);
5424:  URL_SCHEME_ABOUT','LongInt').SetInt( 17);
5425:  URL_SCHEME_RES','LongInt').SetInt( 18);
5426:  URL_SCHEME_MAXVALUE','LongInt').SetInt( 19);
5427:  URL_SCHEME', 'Integer');
5428:  URL_PART_NONE','LongInt').SetInt( 0);
5429:  URL_PART_SCHEME','LongInt').SetInt( 1);
5430:  URL_PART_HOSTNAME','LongInt').SetInt( 2);
5431:  URL_PART_USERNAME','LongInt').SetInt( 3);
5432:  URL_PART_PASSWORD','LongInt').SetInt( 4);
5433:  URL_PART_PORT','LongInt').SetInt( 5);
```

```
5434:  URL_PART_QUERY','LongInt').SetInt( 6);
5435:  URL_PART', 'DWORD');
5436:  URLIS_URL','LongInt').SetInt( 0);
5437:  URLIS_OPAQUE','LongInt').SetInt( 1);
5438:  URLIS_NOHISTORY','LongInt').SetInt( 2);
5439:  URLIS_FILEURL','LongInt').SetInt( 3);
5440:  URLIS_APPLIABLE','LongInt').SetInt( 4);
5441:  URLIS_DIRECTORY','LongInt').SetInt( 5);
5442:  URLIS_HASQUERY','LongInt').SetInt( 6);
5443:  TUrlIs', 'DWORD');
5444:  URL_UNESCAPE','LongWord').SetUInt( $10000000);
5445:  URL_ESCAPE_UNSAFE','LongWord').SetUInt( $20000000);
5446:  URL_PLUGGABLE_PROTOCOL','LongWord').SetUInt( $40000000);
5447:  URL_WININET_COMPATIBILITY','LongWord').SetUInt( DWORD ( $80000000 ));
5448:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord').SetUInt( $02000000);
5449:  URL_ESCAPE_SPACES_ONLY','LongWord').SetUInt( $04000000);
5450:  URL_DONT_SIMPLIFY','LongWord').SetUInt( $08000000);
5451:  URL_NO_META','longword').SetUInt( URL_DONT_SIMPLIFY);
5452:  URL_UNESCAPE_INPLACE','LongWord').SetUInt( $00100000);
5453:  URL_CONVERT_IF_DOSPATH','LongWord').SetUInt( $00200000);
5454:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord').SetUInt( $00400000);
5455:  URL_INTERNAL_PATH','LongWord').SetUInt( $00800000);
5456:  URL_FILE_USE_PATHURL','LongWord').SetUInt( $00010000);
5457:  URL_ESCAPE_PERCENT','LongWord').SetUInt( $00001000);
5458:  URL_ESCAPE_SEGMENT_ONLY','LongWord').SetUInt( $00002000);
5459:  URL_PARTFLAG_KEEPSCHEME','LongWord').SetUInt( $00000001);
5460:  URL_APPLY_DEFAULT','LongWord').SetUInt( $00000001);
5461:  URL_APPLY_GUESSSCHEME','LongWord').SetUInt( $00000002);
5462:  URL_APPLY_GUESSFILE','LongWord').SetUInt( $00000004);
5463:  URL_APPLY_FORCEAPPLY','LongWord').SetUInt( $00000008);
5464:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer');
5465:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombined:PChar;out pcchCombined:DWORD;dwFlags:DWORD):
      HRESULT;
5466:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
5467:  Function UrlIsOpaque( pszURL : PChar) : BOOL');
5468:  Function UrlIsNoHistory( pszURL : PChar) : BOOL');
5469:  Function UrlIsFileUrl( pszURL : PChar) : BOOL');
5470:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL');
5471:  Function UrlGetLocation( psz1 : PChar) : PChar');
5472:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT');
5473:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT');
5474:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT');
5475:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT');
5476:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT');
5477:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT');
5478:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT');
5479:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT');
5480:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT');
5481:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT');
5482:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD');
5483:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD');
5484:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD');
5485:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint');
5486:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName : PChar; var pcchValueName : DWORD;
      pdwType : DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint');
5487:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
5488:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD');
5489:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD');
5490:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD');
5491:  SHREGDEL_DEFAULT','LongWord').SetUInt( $00000000);
5492:  SHREGDEL_HKCU','LongWord').SetUInt( $00000001);
5493:  SHREGDEL_HKLM','LongWord').SetUInt( $00000010);
5494:  SHREGDEL_BOTH','LongWord').SetUInt( $00000011);
5495:  SHREGENUM_DEFAULT','LongWord').SetUInt( $00000000);
5496:  SHREGENUM_HKCU','LongWord').SetUInt( $00000001);
5497:  SHREGENUM_HKLM','LongWord').SetUInt( $00000010);
5498:  SHREGENUM_BOTH','LongWord').SetUInt( $00000011);
5499:  SHREGSET_HKCU','LongWord').SetUInt( $00000001);
5500:  SHREGSET_FORCE_HKCU','LongWord').SetUInt( $00000002);
5501:  SHREGSET_HKLM','LongWord').SetUInt( $00000004);
5502:  SHREGSET_FORCE_HKLM','LongWord').SetUInt( $00000008);
5503:  TSHRegDelFlags', 'DWORD');
5504:  TSHRegEnumFlags', 'DWORD');
5505:  HUSKEY', 'THandle');
5506:  ASSOCF_INIT_NOREMAPCLSID','LongWord').SetUInt( $00000001);
5507:  ASSOCF_INIT_BYEXENAME','LongWord').SetUInt( $00000002);
5508:  ASSOCF_OPEN_BYEXENAME','LongWord').SetUInt( $00000002);
5509:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord').SetUInt( $00000004);
5510:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord').SetUInt( $00000008);
5511:  ASSOCF_NOUSERSETTINGS','LongWord').SetUInt( $00000010);
5512:  ASSOCF_NOTRUNCATE','LongWord').SetUInt( $00000020);
5513:  ASSOCF_VERIFY','LongWord').SetUInt( $00000040);
5514:  ASSOCF_REMAPRUNDLL','LongWord').SetUInt( $00000080);
5515:  ASSOCF_NOFIXUPS','LongWord').SetUInt( $00000100);
5516:  ASSOCF_IGNOREBASECLASS','LongWord').SetUInt( $00000200);
5517:  ASSOCF', 'DWORD');
```

```
5518:  ASSOCSTR_COMMAND','LongInt').SetInt( 1);
5519:  ASSOCSTR_EXECUTABLE','LongInt').SetInt( 2);
5520:  ASSOCSTR_FRIENDLYDOCNAME','LongInt').SetInt( 3);
5521:  ASSOCSTR_FRIENDLYAPPNAME','LongInt').SetInt( 4);
5522:  ASSOCSTR_NOOPEN','LongInt').SetInt( 5);
5523:  ASSOCSTR_SHELLNEWVALUE','LongInt').SetInt( 6);
5524:  ASSOCSTR_DDECOMMAND','LongInt').SetInt( 7);
5525:  ASSOCSTR_DDEIFEXEC','LongInt').SetInt( 8);
5526:  ASSOCSTR_DDEAPPLICATION','LongInt').SetInt( 9);
5527:  ASSOCSTR_DDETOPIC','LongInt').SetInt( 10);
5528:  ASSOCSTR_INFOTIP','LongInt').SetInt( 11);
5529:  ASSOCSTR_MAX','LongInt').SetInt( 12);
5530:  ASSOCSTR', 'DWORD');
5531:  ASSOCKEY_SHELLEXECCLASS','LongInt').SetInt( 1);
5532:  ASSOCKEY_APP','LongInt').SetInt( 2);
5533:  ASSOCKEY_CLASS','LongInt').SetInt( 3);
5534:  ASSOCKEY_BASECLASS','LongInt').SetInt( 4);
5535:  ASSOCKEY_MAX','LongInt').SetInt( 5);
5536:  ASSOCKEY', 'DWORD');
5537:  ASSOCDATA_MSIDESCRIPTOR','LongInt').SetInt( 1);
5538:  ASSOCDATA_NOACTIVATEHANDLER','LongInt').SetInt( 2);
5539:  ASSOCDATA_QUERYCLASSSTORE','LongInt').SetInt( 3);
5540:  ASSOCDATA_HASPERUSERASSOC','LongInt').SetInt( 4);
5541:  ASSOCDATA_MAX','LongInt').SetInt( 5);
5542:  ASSOCDATA', 'DWORD');
5543:  ASSOCENUM_NONE','LongInt').SetInt( 0);
5544:  ASSOCENUM', 'DWORD');
5545:  SID_IQueryAssociations','String').SetString( '{c46ca590-3c3f-11d2-bee6-0000f805ca57}');
5546:  SHACF_DEFAULT','LongWord').SetUInt( $00000000);
5547:  SHACF_FILESYSTEM','LongWord').SetUInt( $00000001);
5548:  SHACF_URLHISTORY','LongWord').SetUInt( $00000002);
5549:  SHACF_URLMRU','LongWord').SetUInt( $00000004);
5550:  SHACF_USETAB','LongWord').SetUInt( $00000008);
5551:  SHACF_FILESYS_ONLY','LongWord').SetUInt( $00000010);
5552:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord').SetUInt( $10000000);
5553:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord').SetUInt( $20000000);
5554:  SHACF_AUTOAPPEND_FORCE_ON','LongWord').SetUInt( $40000000);
5555:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord').SetUInt( DWORD ( $80000000 ));
5556:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT');
5557:  Procedure SHSetThreadRef( punk : IUnknown)');
5558:  Procedure SHGetThreadRef( out ppunk : IUnknown)');
5559:  CTF_INSIST','LongWord').SetUInt( $00000001);
5560:  CTF_THREAD_REF','LongWord').SetUInt( $00000002);
5561:  CTF_PROCESS_REF','LongWord').SetUInt( $00000004);
5562:  CTF_COINIT','LongWord').SetUInt( $00000008);
5563:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE');
5564:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)');
5565:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef');
5566:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef');
5567:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH');
5568:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL');
5569:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer');
5570:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer');
5571:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL');
5572:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL');
5573:  Function SetRectEmpty( var lprc : TRect) : BOOL');
5574:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL');
5575:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL');
5576:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL');
5577:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL');
5578:
5579:  Function InitializeFlatSB( hWnd : HWND) : Bool');
5580:  Procedure UninitializeFlatSB( hWnd : HWND)');
5581:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool');
5582:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool');
5583:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word');  //of JvWin32
5584:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word');
5585:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer');
5586:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word');
5587:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word');
5588:
5589:
5590: //  ***************************************** 204 unit uPSI_ShellAPI;
5591: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT');
5592: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL');
5593: Procedure DragFinish( Drop : HDROP)');
5594: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)');
5595: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST');
5596: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST');
5597: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer');
5598: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON');
5599: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON');
5600: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON');
5601: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT');
5602: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD');
5603: Function ExtractIconEx( lpszFile : PChar; nIconIndex : Integer; var phiconLarge, phiconSmall : HICON;
      nIcons : UINT) : UINT');
```

```
5604: Procedure SHFreeNameMappings( hNameMappings : THandle)');
5605:
5606:  DLLVER_PLATFORM_WINDOWS','LongWord').SetUInt( $00000001);
5607:  DLLVER_PLATFORM_NT','LongWord').SetUInt( $00000002);
5608:  DLLVER_MAJOR_MASK','LongWord').SetUInt( Int64 ( $FFFF000000000000 ));
5609:  DLLVER_MINOR_MASK','LongWord').SetUInt( Int64 ( $0000FFFF00000000 ));
5610:  DLLVER_BUILD_MASK','LongWord').SetUInt( Int64 ( $00000000FFFF0000 ));
5611:  DLLVER_QFE_MASK','LongWord').SetUInt( Int64 ( $000000000000FFFF ));
5612:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64');
5613:  Function SimpleXMLEncode( const S : string) : string');
5614:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)');
5615:  Function XMLEncode( const S : string) : string');
5616:  Function XMLDecode( const S : string) : string');
5617:  Function EntityEncode( const S : string) : string');
5618:  Function EntityDecode( const S : string) : string');
5619:  Procedure EnumComPorts( Ports : TStrings)');
5620:  Function StrToBaudRate( Str : string) : TBaudRate');
5621:  Function StrToStopBits( Str : string) : TStopBits');
5622:  Function StrToDataBits( Str : string) : TDataBits');
5623:  Function StrToParity( Str : string) : TParityBits');
5624:  Function StrToFlowControl( Str : string) : TFlowControl');
5625:  Function BaudRateToStr( BaudRate : TBaudRate) : string');
5626:  Function StopBitsToStr( StopBits : TStopBits) : string');
5627:  Function DataBitsToStr( DataBits : TDataBits) : string');
5628:  Function ParityToStr( Parity : TParityBits) : string');
5629:  Function FlowControlToStr( FlowControl : TFlowControl) : string');
5630:  Function ComErrorsToStr( Errors : TComErrors) : String');
5631:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL');
5632:  Function DispatchMessage( const lpMsg : TMsg) : Longint');
5633:  Function TranslateMessage( const lpMsg : TMsg) : BOOL');
5634:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL');
5635:  Function PeekMessage( var lpMsg:TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL');
5636:  Function GetMessagePos : DWORD');
5637:  Function GetMessageTime : Longint');
5638:  Function GetMessageExtraInfo : Longint');
5639:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string');
5640:  Procedure JAddToRecentDocs( const Filename : string)');
5641:  Procedure ClearRecentDocs');
5642:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON');
5643:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string');
5644:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)');
5645:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)');
5646:  Function RecycleFile( FileToRecycle : string) : Boolean');
5647:  Function JCopyFile( FromFile, ToDir : string) : Boolean');
5648:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT');
5649:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType');
5650:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL');
5651:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL');
5652:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL');
5653:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
      lpResumeHandle : DWORD; pszGroupName : LPCSTR) : BOOL');
5654:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
      lpResumeHandle : DWORD; pszGroupName : LPCWSTR) : BOOL');
5655:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
      lpResumeHandle : DWORD; pszGroupName : LPCTSTR) : BOOL');
5656:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL');
5657:
5658: ****************************************** unit uPSI_JclPeImage;
5659:
5660:  Function IsValidPeFile( const FileName : TFileName) : Boolean');
5661: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean');
5662:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean');
5663:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo');
5664:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean');
5665:  Function PeClearCheckSum( const FileName : TFileName) : Boolean');
5666:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean');
5667:  Function PeDoesExportFunction(const FileName:TFileName;const
      FunctionName:string;Options:TJclSmartCompOptions):Boolean;
5668:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean');
5669:  Function PeIsExportFunctionForwarded( const FileName : TFileName; const FunctionName : string; Options :
      TJclSmartCompOptions) : Boolean');
5670:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean');
5671:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
5672:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean');
5673:  Function PeImportedFunctions( const FileName : TFileName; const FunctionsList : TStrings; const
      LibraryName : string; IncludeLibNames : Boolean) : Boolean');
```

```
5674:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean');
5675:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean');
5676:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean');
5677:  Function PeResourceKindNames( const FileName : TFileName; ResourceType: TJclPeResourceKind; const
      NamesList:TStrings) : Boolean');
5678:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean');
5679:  Function PeBorDependedPackages( const FileName : TFileName; PackagesList : TStrings; FullPathName,
      Descriptions : Boolean) : Boolean');
5680:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;');
5681:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;');
5682:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
5683:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders');
5684:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string');
5685:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader');
5686:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
      PImageSectionHeader');
5687:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean');
5688:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer');
5689:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
      ___Pointer;
5690:   SIRegister_TJclPeSectionStream(CL);
5691:   SIRegister_TJclPeMapImgHookItem(CL);
5692:   SIRegister_TJclPeMapImgHooks(CL);
5693:  //Function PeDbgImgNtHeaders( ProcessHandle: THandle; BaseAddress:Pointer;var NtHeaders:TImageNtHeaders):
      Boolean');
5694:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean');
5695:   Type TJclBorUmSymbolKind',(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)');
5696:   TJclBorUmSymbolModifier', ( smQualified, smLinkProc )');
5697:   TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier');
5698:   TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end');
5699:   TJclBorUmResult', ( urOk, urNotMangled, urMicrosoft, urError )');
5700:   TJclPeUmResult', ( umNotMangled, umBorland, umMicrosoft )');
5701:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
      TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;');
5702:  Function PeBorUnmangleName1( const Name : string; var Unmangled : string; var Description :
      TJclBorUmDescription) : TJclBorUmResult;');
5703:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;');
5704:  Function PeBorUnmangleName3( const Name : string) : string;');
5705:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult');
5706:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult');
5707:
5708:
5709: //****************** SysTools uPSI_StSystem; ************************************
5710:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal');
5711:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString');
5712:  Function DeleteVolumeLabel( Drive : Char) : Cardinal');
5713:  //Procedure EnumerateDirectories( const StartDir : AnsiString; FL : TStrings; SubDirs : Boolean;
      IncludeItem : TIncludeItemFunc)');
5714:  //Procedure EnumerateFiles( const StartDir : AnsiString; FL : TStrings; SubDirs : Boolean; IncludeItem :
      TIncludeItemFunc)');
5715:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal');
5716:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean');
5717:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec');
5718:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer');
5719:  Function FlushOsBuffers( Handle : Integer) : Boolean');
5720:  Function GetCurrentUser : AnsiString');
5721:  Function GetDiskClass( Drive : Char) : DiskClass');
5722:  Function GetDiskInfo(Drive:Char;var ClustersAvailable,TotalClusters,BytesPerSector,
      SectorsPerCluster:Cardinal):Boolean;
5723:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double; var
      DiskSize:Double):Boolean;
5724:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
      DiskSize:Comp):Boolean;
5725:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime');
5726:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime');
5727:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime');
5728:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString');
5729:  Function GetLongPath( const APath : AnsiString) : AnsiString');
5730:  Function GetMachineName : AnsiString');
5731:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal');
5732:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString');
5733:  Function GetShortPath( const APath : AnsiString) : AnsiString');
5734:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString');
5735:  Function GetTempFolder( aForceSlash : boolean) : AnsiString');
5736:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString');
5737:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString');
5738:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec');
5739:  Function StIsDirectory( const DirName : AnsiString) : Boolean');
5740:  Function IsDirectoryEmpty( const S : AnsiString) : Integer');
5741:  Function IsDriveReady( Drive : Char) : Boolean');
5742:  Function IsFile( const FileName : AnsiString) : Boolean');
5743:  Function IsFileArchive( const S : AnsiString) : Integer');
5744:  Function IsFileHidden( const S : AnsiString) : Integer');
5745:  Function IsFileReadOnly( const S : AnsiString) : Integer');
5746:  Function IsFileSystem( const S : AnsiString) : Integer');
5747:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec');
```

```
5748:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal');
5749:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean');
5750:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal');
5751:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)');
5752:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt');
5753:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint');
5754:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec');
5755:  Function ValidDrive( Drive : Char) : Boolean');
5756:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal');
5757:
5758: //****************************unit uPSI_JclLANMan;****************************************
5759: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean');
5760:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
      const PasswordNeverExpires : Boolean) : Boolean');
5761:  Function DeleteAccount( const Servername, Username : string) : Boolean');
5762:  Function DeleteLocalAccount( Username : string) : Boolean');
5763:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean');
5764:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean');
5765:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean');
5766:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean');
5767:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean');
5768:  Function LocalGroupExists( const Group : string) : Boolean');
5769:  Function GlobalGroupExists( const Server, Group : string) : Boolean');
5770:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean');
5771:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string');
5772:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)');
5773:  Function IsLocalAccount( const AccountName : string) : Boolean');
5774:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer');
5775:  Function GetRandomString( NumChar : cardinal) : string');
5776:
5777: //****************************unit uPSI_cUtils;****************************************
5778: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )');
5779: Function cIsWinNT : boolean');
5780:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
5781:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle');
5782:  Function cRunAndGetOutput( Cmd, WorkDir : string; ErrFunc : TErrFunc; LineOutputFunc : TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string');
5783:  Function cGetShortName( FileName : string) : string');
5784:  Procedure cShowError( Msg : String)');
5785:  Function cCommaStrToStr( s : string; formatstr : string) : string');
5786:  Function cIncludeQuoteIfSpaces( s : string) : string');
5787:  Function cIncludeQuoteIfNeeded( s : string) : string');
5788:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)');
5789:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
5790:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;');
5791:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;');
5792:  Function cCodeInstoStr( s : string) : string');
5793:  Function cStrtoCodeIns( s : string) : string');
5794:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)');
5795:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string');
5796:  Procedure cStrtoPoint( var pt : TPoint; value : string)');
5797:  Function cPointtoStr( const pt : TPoint) : string');
5798:  Function cListtoStr( const List : TStrings) : string');
5799:  Function ListtoStr( const List : TStrings) : string');
5800:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)');
5801:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)');
5802:  Function cGetFileTyp( const FileName : string) : TUnitType');
5803:  Function cGetExTyp( const FileName : string) : TExUnitType');
5804:  Procedure cSetPath( Add : string; const UseOriginal : boolean)');
5805:  Function cExpandFileto( const FileName : string; const BasePath : string) : string');
5806:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean');
5807:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)');
5808:  Function cGetLastPos( const SubStr : string; const S : string) : integer');
5809:  Function cGenMakePath( FileName : String) : String;');
5810:  Function cGenMakePath2( FileName : String) : String');
5811:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;');
5812:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String');
5813:  Function cCalcMod( Count : Integer) : Integer');
5814:  Function cGetVersionString( FileName : string) : string');
5815:  Function cCheckChangeDir( var Dir : string) : boolean');
5816:  Function cGetAssociatedProgram(const Extension: string; var Filename,Description: string): boolean');
5817:  Function cIsNumeric( s : string) : boolean');
5818:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)');
5819:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string');
5820:  Function GetFileTyp( const FileName : string) : TUnitType');
5821:  Function Atoi(const aStr: string): integer');
5822:  Function Itoa(const aint: integer): string');
5823:
5824:
5825:
5826: //****************************unit uPSI_BoldUtils;****************************************
5827:  Function CharCount( c : char; const s : string) : integer');
5828:  Function BoldNamesEqual( const name1, name2 : string) : Boolean');
5829:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)');
```

```
5830:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String');
5831:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string');
5832:  Function BoldTrim( const S : string) : string');
5833:  Function BoldIsPrefix( const S, Prefix : string) : Boolean');
5834:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean');
5835:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean');
5836:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean');
5837:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean');
5838:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer');
5839:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)');
5840:  Function CapitalisedToSpaced( Capitalised : String) : String');
5841:  Function SpacedToCapitalised( Spaced : String) : String');
5842:  Function BooleanToString( BoolValue : Boolean) : String');
5843:  Function StringToBoolean( StrValue : String) : Boolean');
5844:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer');
5845:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer');
5846:  Function StringListToVarArray( List : TStringList) : variant');
5847:  Function IsLocalMachine( const Machinename : WideString) : Boolean');
5848:  Function GetComputerNameStr : string');
5849:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer');
5850:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const
      DateSeparatorChar:char):TDateTime');
5851:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
5852:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
5853:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
5854:  Procedure EnsureTrailing( var Str : String; ch : char)');
5855:  Function BoldDirectoryExists( const Name : string) : Boolean');
5856:  Function BoldForceDirectories( Dir : string) : Boolean');
5857:  Function BoldRootRegistryKey : string');
5858:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string');
5859:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer');
5860:  Function LogicalAnd( A, B : Integer) : Boolean');
5861:  record TByHandleFileInformation dwFileAttributes : DWORD; '
5862:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
5863:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
5864:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end');
5865:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
5866:  Function IsFirstInstance : Boolean');
5867:  Procedure ActivateFirst( AString : PChar)');
5868:  Procedure ActivateFirstCommandLine');
5869:  function MakeAckPkt(const BlockNumber: Word): string;
5870:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer; const ErrNumber: Word;
      ErrorString: string); overload;
5871:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
5872:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
5873:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
5874:  function IdStrToWord(const Value: String): Word;
5875:  function IdWordToStr(const Value: Word): WordStr;
5876:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean');
5877:  Function CPUFeatures : TCPUFeatures');
5878:
5879: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
5880: begin
5881:   AddTypeS('TXRTLBitIndex', 'Integer');
5882:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal');
5883:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean');
5884:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal');
5885:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal');
5886:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal');
5887:  Function XRTLSwapHiLo16( X : Word) : Word');
5888:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal');
5889:  Function XRTLSwapHiLo64( X : Int64) : Int64');
5890:  Function XRTLROL32( A, S : Cardinal) : Cardinal');
5891:  Function XRTLROR32( A, S : Cardinal) : Cardinal');
5892:  Function XRTLROL16( A : Word; S : Cardinal) : Word');
5893:  Function XRTLROR16( A : Word; S : Cardinal) : Word');
5894:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte');
5895:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte');
5896:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)');
5897:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)');
5898:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)');
5899:  Function XRTLPopulation( A : Cardinal) : Cardinal');
5900: end;
5901:
5902: Function XRTLURLDecode( const ASrc : WideString) : WideString');
5903: Function XRTLURLEncode( const ASrc : WideString) : WideString');
5904: Function XRTLURINormalize( const AURI : WideString) : WideString');
5905: Procedure XRTLURIParse( const AURI: WideString; var VProtocol, VHost, VPath, VDocument, VPort, VBookmark,
      VUserName, VPassword : WideString)');
5906: Function XRTLExtractLongPathName(APath: string): string;
5907:
5908: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
5909: begin
5910:   AddTypeS('Int8', 'ShortInt');
5911:   AddTypeS('Int16', 'SmallInt');
```

```
5912:   AddTypeS('Int32', 'LongInt');
5913:   AddTypeS('UInt8', 'Byte');
5914:   AddTypeS('UInt16', 'Word');
5915:   AddTypeS('UInt32', 'LongWord');
5916:   AddTypeS('UInt64', 'Int64');
5917:   AddTypeS('Word8', 'UInt8');
5918:   AddTypeS('Word16', 'UInt16');
5919:   AddTypeS('Word32', 'UInt32');
5920:   AddTypeS('Word64', 'UInt64');
5921:   AddTypeS('LargeInt', 'Int64');
5922:   AddTypeS('NativeInt', 'Integer');
5923:   AddTypeS('NativeUInt', 'Cardinal');
5924:   AddConstantN('BitsPerByte','LongInt').SetInt( 8);
5925:   AddConstantN('BitsPerWord','LongInt').SetInt( 16);
5926:   AddConstantN('BitsPerLongWord','LongInt').SetInt( 32);
5927:  //AddConstantN('BitsPerCardinal','LongInt').SetInt( BytesPerCardinal * 8);
5928:  //AddConstantN('BitsPerNativeWord','LongInt').SetInt( BytesPerNativeWord * 8);
5929:  Function MinI( const A, B : Integer) : Integer');
5930:  Function MaxI( const A, B : Integer) : Integer');
5931:  Function MinC( const A, B : Cardinal) : Cardinal');
5932:  Function MaxC( const A, B : Cardinal) : Cardinal');
5933:  Function SumClipI( const A, I : Integer) : Integer');
5934:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal');
5935:  Function InByteRange( const A : Int64) : Boolean');
5936:  Function InWordRange( const A : Int64) : Boolean');
5937:  Function InLongWordRange( const A : Int64) : Boolean');
5938:  Function InShortIntRange( const A : Int64) : Boolean');
5939:  Function InSmallIntRange( const A : Int64) : Boolean');
5940:  Function InLongIntRange( const A : Int64) : Boolean');
5941:   AddTypeS('Bool8', 'ByteBool');
5942:   AddTypeS('Bool16', 'WordBool');
5943:   AddTypeS('Bool32', 'LongBool');
5944:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )');
5945:   AddTypeS('TCompareResultSet', 'set of TCompareResult');
5946:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult');
5947:  ConstantN('MinSingle','Single').setExtended( 1.5E-45);
5948:  ConstantN('MaxSingle','Single').setExtended( 3.4E+38);
5949:  ConstantN('MinDouble','Double').setExtended( 5.0E-324);
5950:  ConstantN('MaxDouble','Double').setExtended( 1.7E+308);
5951:  ConstantN('MinExtended','Extended').setExtended(3.4E-4932);
5952:  ConstantN('MaxExtended','Extended').setExtended(1.1E+4932);
5953:  ConstantN('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
5954:  ConstantN('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
5955:  Function MinF( const A, B : Float) : Float');
5956:  Function MaxF( const A, B : Float) : Float');
5957:  Function ClipF( const Value : Float; const Low, High : Float) : Float');
5958:  Function InSingleRange( const A : Float) : Boolean');
5959:  Function InDoubleRange( const A : Float) : Boolean');
5960:  Function InCurrencyRange( const A : Float) : Boolean;');
5961:  Function InCurrencyRange1( const A : Int64) : Boolean;');
5962:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean');
5963:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean');
5964:  Function FloatIsInfinity( const A : Extended) : Boolean');
5965:  Function FloatIsNaN( const A : Extended) : Boolean');
5966:  AddConstantN('SingleCompareDelta','Extended').setExtended( 1.0E-34);
5967:  AddConstantN('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
5968:  AddConstantN('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
5969:  AddConstantN('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
5970:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean');
5971:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean');
5972:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean');
5973:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult');
5974:  AddConstantN('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
5975:  AddConstantN('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
5976:  AddConstantN('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
5977:  AddConstantN('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
5978:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean');
5979:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult');
5980:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord');
5981:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord');
5982:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean');
5983:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord');
5984:  Function cIsHighBitSet( const Value : LongWord) : Boolean');
5985:  Function SetBitScanForward( const Value : LongWord) : Integer;');
5986:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;');
5987:  Function SetBitScanReverse( const Value : LongWord) : Integer;');
5988:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;');
5989:  Function ClearBitScanForward( const Value : LongWord) : Integer;');
5990:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;');
5991:  Function ClearBitScanReverse( const Value : LongWord) : Integer;');
5992:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;');
5993:  Function cReverseBits( const Value : LongWord) : LongWord;');
5994:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;');
5995:  Function cSwapEndian( const Value : LongWord) : LongWord');
5996:  Function cTwosComplement( const Value : LongWord) : LongWord');
5997:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word');
```

```
5998:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord');
5999:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word');
6000:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord');
6001:  Function cBitCount( const Value : LongWord) : LongWord');
6002:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean');
6003:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord');
6004:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord');
6005:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord');
6006:  Function SetBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord');
6007:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord');
6008:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord');
6009:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean');
6010:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean');
6011: //  AddTypeS('CharSet', 'set of AnsiChar');
6012:   AddTypeS('CharSet', 'set of Char');       //!!!
6013:   AddTypeS('AnsiCharSet', 'TCharSet');
6014:   AddTypeS('ByteSet', 'set of Byte');
6015:   AddTypeS('AnsiChar', 'Char');
6016:    // Function AsCharSet( const C : array of AnsiChar) : CharSet');
6017:  Function AsByteSet( const C : array of Byte) : ByteSet');
6018:  Procedure ComplementChar( var C : CharSet; const Ch : Char)');
6019:  Procedure ClearCharSet( var C : CharSet)');
6020:  Procedure FillCharSet( var C : CharSet)');
6021:  Procedure ComplementCharSet( var C : CharSet)');
6022:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)');
6023:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)');
6024:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)');
6025:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)');
6026:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)');
6027:  Function IsSubSet( const A, B : CharSet) : Boolean');
6028:  Function IsEqual( const A, B : CharSet) : Boolean');
6029:  Function IsEmpty( const C : CharSet) : Boolean');
6030:  Function IsComplete( const C : CharSet) : Boolean');
6031:  Function cCharCount( const C : CharSet) : Integer');
6032:  Procedure ConvertCaseInsensitive( var C : CharSet)');
6033:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet');
6034:  Function IntRangeLength( const Low, High : Integer) : Int64');
6035:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean');
6036:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean');
6037:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean');
6038:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean');
6039:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
6040:  Function CardRangeLength( const Low, High : Cardinal) : Int64');
6041:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean');
6042:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean');
6043:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean');
6044:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean');
6045:  Function CardRangeIncludeElementRange(var Low,High:Cardinal;const LowElement,
      HighElement:Cardinal):Boolean;
6046:   AddTypeS('UnicodeChar', 'WideChar');
6047:  Function Compare( const I1, I2 : Boolean) : TCompareResult;');
6048:  Function Compare1( const I1, I2 : Integer) : TCompareResult;');
6049:  Function Compare2( const I1, I2 : Int64) : TCompareResult;');
6050:  Function Compare3( const I1, I2 : Extended) : TCompareResult;');
6051:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult');
6052:  Function CompareW( const I1, I2 : WideString) : TCompareResult');
6053:  Function cSgn( const A : LongInt) : Integer;');
6054:  Function cSgn1( const A : Int64) : Integer;');
6055:  Function cSgn2( const A : Extended) : Integer;');
6056:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )');
6057:  Function AnsiCharToInt( const A : AnsiChar) : Integer');
6058:  Function WideCharToInt( const A : WideChar) : Integer');
6059:  Function CharToInt( const A : Char) : Integer');
6060:  Function IntToAnsiChar( const A : Integer) : AnsiChar');
6061:  Function IntToWideChar( const A : Integer) : WideChar');
6062:  Function IntToChar( const A : Integer) : Char');
6063:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean');
6064:  Function IsHexWideChar( const Ch : WideChar) : Boolean');
6065:  Function IsHexChar( const Ch : Char) : Boolean');
6066:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer');
6067:  Function HexWideCharToInt( const A : WideChar) : Integer');
6068:  Function HexCharToInt( const A : Char) : Integer');
6069:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar');
6070:  Function IntToUpperHexWideChar( const A : Integer) : WideChar');
6071:  Function IntToUpperHexChar( const A : Integer) : Char');
6072:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar');
6073:  Function IntToLowerHexWideChar( const A : Integer) : WideChar');
6074:  Function IntToLowerHexChar( const A : Integer) : Char');
6075:  Function IntToStringA( const A : Int64) : AnsiString');
6076:  Function IntToStringW( const A : Int64) : WideString');
6077:  Function IntToString( const A : Int64) : String');
6078:  Function UIntToStringA( const A : NativeUInt) : AnsiString');
6079:  Function UIntToStringW( const A : NativeUInt) : WideString');
6080:  Function UIntToString( const A : NativeUInt) : String');
6081:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString');
6082:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString');
```

```
6083:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString');
6084:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String');
6085:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
6086:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
6087:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String');
6088:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString');
6089:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString');
6090:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String');
6091:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString');
6092:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString');
6093:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String');
6094:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean');
6095:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean');
6096:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean');
6097:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64');
6098:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64');
6099:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64');
6100:  Function StringToInt64A( const S : AnsiString) : Int64');
6101:  Function StringToInt64W( const S : WideString) : Int64');
6102:  Function StringToInt64( const S : String) : Int64');
6103:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean');
6104:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean');
6105:  Function TryStringToInt( const S : String; out A : Integer) : Boolean');
6106:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer');
6107:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer');
6108:  Function StringToIntDef( const S : String; const Default : Integer) : Integer');
6109:  Function StringToIntA( const S : AnsiString) : Integer');
6110:  Function StringToIntW( const S : WideString) : Integer');
6111:  Function StringToInt( const S : String) : Integer');
6112:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean');
6113:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean');
6114:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean');
6115:  Function StringToLongWordA( const S : AnsiString) : LongWord');
6116:  Function StringToLongWordW( const S : WideString) : LongWord');
6117:  Function StringToLongWord( const S : String) : LongWord');
6118:  Function HexToUIntA( const S : AnsiString) : NativeUInt');
6119:  Function HexToUIntW( const S : WideString) : NativeUInt');
6120:  Function HexToUInt( const S : String) : NativeUInt');
6121:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean');
6122:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean');
6123:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean');
6124:  Function HexToLongWordA( const S : AnsiString) : LongWord');
6125:  Function HexToLongWordW( const S : WideString) : LongWord');
6126:  Function HexToLongWord( const S : String) : LongWord');
6127:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean');
6128:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean');
6129:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean');
6130:  Function OctToLongWordA( const S : AnsiString) : LongWord');
6131:  Function OctToLongWordW( const S : WideString) : LongWord');
6132:  Function OctToLongWord( const S : String) : LongWord');
6133:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean');
6134:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean');
6135:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean');
6136:  Function BinToLongWordA( const S : AnsiString) : LongWord');
6137:  Function BinToLongWordW( const S : WideString) : LongWord');
6138:  Function BinToLongWord( const S : String) : LongWord');
6139:  Function FloatToStringA( const A : Extended) : AnsiString');
6140:  Function FloatToStringW( const A : Extended) : WideString');
6141:  Function FloatToString( const A : Extended) : String');
6142:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean');
6143:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean');
6144:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean');
6145:  Function StringToFloatA( const A : AnsiString) : Extended');
6146:  Function StringToFloatW( const A : WideString) : Extended');
6147:  Function StringToFloat( const A : String) : Extended');
6148:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended');
6149:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended');
6150:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended');
6151:  Function EncodeBase64( const S, Alphabet : AnsiString; const Pad : Boolean; const PadMultiple : Integer;
     const PadChar : AnsiChar) : AnsiString');
6152:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString');
6153:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/');
6154:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
6155:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
6156:  Const('CCHARSET','String').SetString(b64_XXEncode);
6157:  Const('CHEXSET','String').SetString('0123456789ABCDEF');
6158:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString');
6159:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString');
6160:  Function UUDecode( const S : AnsiString) : AnsiString');
6161:  Function XXDecode( const S : AnsiString) : AnsiString');
6162:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString');
6163:  Function InterfaceToStrA( const I : IInterface) : AnsiString');
6164:  Function InterfaceToStrW( const I : IInterface) : WideString');
6165:  Function InterfaceToStr( const I : IInterface) : String');
6166:  Function ObjectClassName( const O : TObject) : String');
6167:  Function ClassClassName( const C : TClass) : String');
```

```
6168:  Function ObjectToStr( const O : TObject) : String');
6169:  Function ObjectToString( const O : TObject) : String');
6170:  Function CharSetToStr( const C : CharSet) : AnsiString');
6171:  Function StrToCharSet( const S : AnsiString) : CharSet');
6172:  Function HashStrA( const S : AnsiString; const Index : Integer; const Count : Integer; const
       AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord');
6173:  Function HashStrW(const S:WideString; const Index:Integer;const Count:Integer; const
       AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord');
6174:  Function HashStr( const S : String; const Index : Integer; const Count : Integer; const
       AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord');
6175:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord');
6176:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord');
6177:  AddConstantN('Bytes1KB','LongInt').SetInt( 1024);
6178:   SIRegister_IInterface(CL);
6179:  Procedure SelfTestCFundamentUtils');
6180:
6181: Function CreateSchedule : IJclSchedule');
6182: Function NullStamp : TTimeStamp');
6183: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64');
6184: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean');
6185: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean');
6186:
6187: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
6188: begin
6189:  AddTypeS('TFunc', 'function(X : Float) : Float;');
6190: Function InitGraphics( Width, Height : Integer) : Boolean');
6191: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)');
6192: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)');
6193: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)');
6194: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)');
6195: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)');
6196: Procedure SetGraphTitle( Title : String)');
6197: Procedure SetOxTitle( Title : String)');
6198: Procedure SetOyTitle( Title : String)');
6199: Function GetGraphTitle : String');
6200: Function GetOxTitle : String');
6201: Function GetOyTitle : String');
6202: Procedure PlotOxAxis( Canvas : TCanvas)');
6203: Procedure PlotOyAxis( Canvas : TCanvas)');
6204: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)');
6205: Procedure WriteGraphTitle( Canvas : TCanvas)');
6206: Function SetMaxCurv( NCurv : Byte) : Boolean');
6207: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)');
6208: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)');
6209: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)');
6210: Procedure SetCurvStep( CurvIndex, Step : Integer)');
6211: Function GetMaxCurv : Byte');
6212: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)');
6213: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
6214: Function GetCurvLegend( CurvIndex : Integer) : String');
6215: Function GetCurvStep( CurvIndex : Integer) : Integer');
6216: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)');
6217: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)');
6218: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)');
6219: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)');
6220: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)');
6221: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)');
6222: Function Xpixel( X : Float) : Integer');
6223: Function Ypixel( Y : Float) : Integer');
6224: Function Xuser( X : Integer) : Float');
6225: Function Yuser( Y : Integer) : Float');
6226: end;
6227:
6228: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)');
6229: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)');
6230: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)');
6231: Procedure FFT_Integer_Cleanup');
6232: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector; var FT : Complex)');
6233:
6234: //************************************************************************unit uPSI_umath;
6235:  Function uExpo( X : Float) : Float');
6236:  Function uExp2( X : Float) : Float');
6237:  Function uExp10( X : Float) : Float');
6238:  Function uLog( X : Float) : Float');
6239:  Function uLog2( X : Float) : Float');
6240:  Function uLog10( X : Float) : Float');
6241:  Function uLogA( X, A : Float) : Float');
6242:  Function uIntPower( X : Float; N : Integer): Float');
6243:  Function uPower( X, Y : Float) : Float');
6244:  Function SgnGamma( X : Float) : Integer');
6245:  Function Stirling( X : Float) : Float');
6246:  Function StirLog( X : Float) : Float');
6247:  Function Gamma( X : Float) : Float');
6248:  Function LnGamma( X : Float) : Float');
6249:
6250: procedure SIRegister_urandom(CL: TPSPascalCompiler);
```

```
6251: begin
6252: type RNG_Type =
6253:   (RNG_MWC,      { Multiply-With-Carry }
6254:    RNG_MT,       { Mersenne Twister }
6255:    RNG_UVAG);    { Universal Virtual Array Generator }
6256:  Procedure SetRNG( RNG : RNG_Type)');
6257:  Procedure InitGen( Seed : RNG_IntType)');
6258:  Function IRanGen : RNG_IntType');
6259:  Function IRanGen31 : RNG_IntType');
6260:  Function RanGen1 : Float');
6261:  Function RanGen2 : Float');
6262:  Function RanGen3 : Float');
6263:  Function RanGen53 : Float');
6264: end;
6265:
6266: //  Optimization by Simulated Annealing
6267: procedure SIRegister_usimann(CL: TPSPascalCompiler);
6268: begin
6269:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)');
6270:  Procedure SA_CreateLogFile( FileName : String)');
6271:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
6272: end;
6273:
6274: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
6275: begin
6276:  Procedure InitUVAGbyString( KeyPhrase : string)');
6277:  Procedure InitUVAG( Seed : RNG_IntType)');
6278:  Function IRanUVAG : RNG_IntType');
6279: end;
6280:
6281: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
6282: begin
6283:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)');
6284:  Procedure GA_CreateLogFile( LogFileName : String)');
6285:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
6286: end;
6287:
6288:  CL.AddTypeS('TVector', 'array of Float');
6289: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
6290: begin
6291:  Procedure QSort( X : TVector; Lb, Ub : Integer)');
6292:  Procedure DQSort( X : TVector; Lb, Ub : Integer)');
6293: end;
6294:
6295: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
6296: begin
6297:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)');
6298:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)');
6299: end;
6300:
6301:
6302: //************************************ PaintFX*************************
6303: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
6304: begin
6305:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
6306:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
6307:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)');
6308:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)');
6309:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)');
6310:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)');
6311:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)');
6312:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)');
6313:     Procedure Turn( Src, Dst : TBitmap)');
6314:     Procedure TurnRight( Src, Dst : TBitmap)');
6315:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)');
6316:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)');
6317:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)');
6318:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)');
6319:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)');
6320:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)');
6321:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)');
6322:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)');
6323:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)');
6324:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)');
6325:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)');
6326:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)');
6327:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)');
6328:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)');
6329:     Procedure Emboss( var Bmp : TBitmap)');
6330:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)');
6331:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)');
6332:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)');
6333:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)');
6334:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)');
6335:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)');
6336:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)');
```

```
6337:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)');
6338:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)');
6339:     Procedure QuartoOpaque( Src, Dst : TBitmap)');
6340:     Procedure SemiOpaque( Src, Dst : TBitmap)');
6341:     Procedure ShadowDownLeft( const Dst : TBitmap)');
6342:     Procedure ShadowDownRight( const Dst : TBitmap)');
6343:     Procedure ShadowUpLeft( const Dst : TBitmap)');
6344:     Procedure ShadowUpRight( const Dst : TBitmap)');
6345:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)');
6346:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)');
6347:     Procedure FlipRight( const Dst : TBitmap)');
6348:     Procedure FlipDown( const Dst : TBitmap)');
6349:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)');
6350:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)');
6351:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)');
6352:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)');
6353:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)');
6354:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)');
6355:     Procedure SmoothResize( var Src, Dst : TBitmap)');
6356:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)');
6357:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)');
6358:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)');
6359:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)');
6360:     Procedure GrayScale( const Dst : TBitmap)');
6361:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)');
6362:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)');
6363:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)');
6364:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)');
6365:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)');
6366:     Procedure Spray( const Dst : TBitmap; Amount : Integer)');
6367:     Procedure AntiAlias( const Dst : TBitmap)');
6368:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)');
6369:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)');
6370:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)');
6371:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)');
6372:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)');
6373:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)');
6374:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)');
6375:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)');
6376:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)');
6377:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)');
6378:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)');
6379:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)');
6380:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)');
6381:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)');
6382:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)');
6383:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)');
6384:     Procedure Invert( Src : TBitmap)');
6385:     Procedure MirrorRight( Src : TBitmap)');
6386:     Procedure MirrorDown( Src : TBitmap)');
6387:   end;
6388: end;
6389:
6390: (*----------------------------------------------------------------------------*)
6391: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
6392: begin
6393:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
6394:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
6395:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )');
6396:   SIRegister_TJvPaintFX(CL);
6397:  Function SplineFilter( Value : Single) : Single');
6398:  Function BellFilter( Value : Single) : Single');
6399:  Function TriangleFilter( Value : Single) : Single');
6400:  Function BoxFilter( Value : Single) : Single');
6401:  Function HermiteFilter( Value : Single) : Single');
6402:  Function Lanczos3Filter( Value : Single) : Single');
6403:  Function MitchellFilter( Value : Single) : Single');
6404: end;
6405:
6406:
6407: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
6408: begin
6409: AddClassN(FindClass('TOBJECT'),'EDateTime');
6410: Function DatePart( const D : TDateTime) : Integer');
6411: Function TimePart( const D : TDateTime) : Double');
6412: Function Century( const D : TDateTime) : Word');
6413: Function Year( const D : TDateTime) : Word');
6414: Function Month( const D : TDateTime) : Word');
6415: Function Day( const D : TDateTime) : Word');
6416: Function Hour( const D : TDateTime) : Word');
6417: Function Minute( const D : TDateTime) : Word');
6418: Function Second( const D : TDateTime) : Word');
6419: Function Millisecond( const D : TDateTime) : Word');
6420:  AddConstantN('OneDay','Extended').setExtended( 1.0);
6421:  AddConstantN('OneHour','Extended').SetExtended( OneDay / 24);
6422:  AddConstantN('OneMinute','Extended').SetExtended( OneHour / 60);
```

```
6423:  AddConstantN('OneSecond','Extended').SetExtended( OneMinute / 60);
6424:  AddConstantN('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
6425:  AddConstantN('OneWeek','Extended').SetExtended( OneDay * 7);
6426:  AddConstantN('HoursPerDay','Extended').SetExtended( 24);
6427:  AddConstantN('MinutesPerHour','Extended').SetExtended( 60);
6428:  AddConstantN('SecondsPerMinute','Extended').SetExtended( 60);
6429: Procedure SetYear( var D : TDateTime; const Year : Word)');
6430: Procedure SetMonth( var D : TDateTime; const Month : Word)');
6431: Procedure SetDay( var D : TDateTime; const Day : Word)');
6432: Procedure SetHour( var D : TDateTime; const Hour : Word)');
6433: Procedure SetMinute( var D : TDateTime; const Minute : Word)');
6434: Procedure SetSecond( var D : TDateTime; const Second : Word)');
6435: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)');
6436: Function IsEqual( const D1, D2 : TDateTime) : Boolean;');
6437: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word) : Boolean;');
6438: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word) : Boolean;');
6439: Function IsAM( const D : TDateTime) : Boolean');
6440: Function IsPM( const D : TDateTime) : Boolean');
6441: Function IsMidnight( const D : TDateTime) : Boolean');
6442: Function IsNoon( const D : TDateTime) : Boolean');
6443: Function IsSunday( const D : TDateTime) : Boolean');
6444: Function IsMonday( const D : TDateTime) : Boolean');
6445: Function IsTuesday( const D : TDateTime) : Boolean');
6446: Function IsWedneday( const D : TDateTime) : Boolean');
6447: Function IsThursday( const D : TDateTime) : Boolean');
6448: Function IsFriday( const D : TDateTime) : Boolean');
6449: Function IsSaturday( const D : TDateTime) : Boolean');
6450: Function IsWeekend( const D : TDateTime) : Boolean');
6451: Function Noon( const D : TDateTime) : TDateTime');
6452: Function Midnight( const D : TDateTime) : TDateTime');
6453: Function FirstDayOfMonth( const D : TDateTime) : TDateTime');
6454: Function LastDayOfMonth( const D : TDateTime) : TDateTime');
6455: Function NextWorkday( const D : TDateTime) : TDateTime');
6456: Function PreviousWorkday( const D : TDateTime) : TDateTime');
6457: Function FirstDayOfYear( const D : TDateTime) : TDateTime');
6458: Function LastDayOfYear( const D : TDateTime) : TDateTime');
6459: Function EasterSunday( const Year : Word) : TDateTime');
6460: Function GoodFriday( const Year : Word) : TDateTime');
6461: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime');
6462: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime');
6463: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime');
6464: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime');
6465: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime');
6466: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime');
6467: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime');
6468: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime');
6469: Function DayOfYear( const Ye, Mo, Da : Word) : Integer');
6470: Function DayOfYear( const D : TDateTime) : Integer');
6471: Function DaysInMonth( const Ye, Mo : Word) : Integer');
6472: Function DaysInMonth( const D : TDateTime) : Integer');
6473: Function DaysInYear( const Ye : Word) : Integer');
6474: Function DaysInYearDate( const D : TDateTime) : Integer');
6475: Function WeekNumber( const D : TDateTime) : Integer');
6476: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime');
6477: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)');
6478: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64');
6479: Function DiffSeconds( const D1, D2 : TDateTime) : Integer');
6480: Function DiffMinutes( const D1, D2 : TDateTime) : Integer');
6481: Function DiffHours( const D1, D2 : TDateTime) : Integer');
6482: Function DiffDays( const D1, D2 : TDateTime) : Integer');
6483: Function DiffWeeks( const D1, D2 : TDateTime) : Integer');
6484: Function DiffMonths( const D1, D2 : TDateTime) : Integer');
6485: Function DiffYears( const D1, D2 : TDateTime) : Integer');
6486: Function GMTBias : Integer');
6487: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime');
6488: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime');
6489: Function NowAsGMTTime : TDateTime');
6490: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString');
6491: Function ISO8601StringToTime( const D : AnsiString) : TDateTime');
6492: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime');
6493: Function DateTimeToANSI( const D : TDateTime) : Integer');
6494: Function ANSIToDateTime( const Julian : Integer) : TDateTime');
6495: Function DateTimeToISOInteger( const D : TDateTime) : Integer');
6496: Function DateTimeToISOString( const D : TDateTime) : AnsiString');
6497: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime');
6498: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer');
6499: Function DateTimeAsElapsedTime( const D : TDateTime; const IncludeMilliseconds : Boolean): AnsiString');
6500: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime');
6501: Function DateTimeToUnixTime( const D : TDateTime) : LongWord');
6502: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString');
6503: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString');
6504: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString');
6505: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString');
6506: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString');
6507: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString');
6508: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString');
```

```
6509: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString');
6510: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer');
6511: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer');
6512: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer');
6513: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer');
6514: Function EnglishShortMonthA( const S : AnsiString) : Integer');
6515: Function EnglishShortMonthU( const S : UnicodeString) : Integer');
6516: Function EnglishLongMonthA( const S : AnsiString) : Integer');
6517: Function EnglishLongMonthU( const S : UnicodeString) : Integer');
6518: Function RFC850DayOfWeekA( const S : AnsiString) : Integer');
6519: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer');
6520: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer');
6521: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer');
6522: Function RFCMonthA( const S : AnsiString) : Word');
6523: Function RFCMonthU( const S : UnicodeString) : Word');
6524: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds : Boolean) : AnsiString');
6525: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds : Boolean) : UnicodeString');
6526: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek: Boolean): AnsiString');
6527: Function GMTDateTimeToRFC1123DateTimeU( const D : TDateTime; const IncludeDayOfWeek : Boolean):
      UnicodeString');
6528: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString');
6529: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString');
6530: Function NowAsRFCDateTimeA : AnsiString');
6531: Function NowAsRFCDateTimeU : UnicodeString');
6532: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime');
6533: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime');
6534: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer');
6535: Function TimePeriodStr( const D : TDateTime) : AnsiString');
6536: Procedure SelfTest');
6537: end;
6538: //****************************************CFileUtils
6539: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean');
6540: Function PathHasDriveLetter( const Path : String) : Boolean');
6541: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean');
6542: Function PathIsDriveLetter( const Path : String) : Boolean');
6543: Function PathIsDriveRootA( const Path : AnsiString) : Boolean');
6544: Function PathIsDriveRoot( const Path : String) : Boolean');
6545: Function PathIsRootA( const Path : AnsiString) : Boolean');
6546: Function PathIsRoot( const Path : String) : Boolean');
6547: Function PathIsUNCPathA( const Path : AnsiString) : Boolean');
6548: Function PathIsUNCPath( const Path : String) : Boolean');
6549: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean');
6550: Function PathIsAbsolute( const Path : String) : Boolean');
6551: Function PathIsDirectoryA( const Path : AnsiString) : Boolean');
6552: Function PathIsDirectory( const Path : String) : Boolean');
6553: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString');
6554: Function PathInclSuffix( const Path : String; const PathSep : Char) : String');
6555: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString');
6556: Function PathExclSuffix( const Path : String; const PathSep : Char) : String');
6557: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)');
6558: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)');
6559: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)');
6560: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)');
6561: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString');
6562: Function PathCanonical( const Path : String; const PathSep : Char) : String');
6563: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString');
6564: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String');
6565: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString');
6566: Function PathLeftElement( const Path : String; const PathSep : Char) : String');
6567: Procedure PathSplitLeftElementA(const Path:AnsiString;var LeftElement,RightPath:AnsiString;const
      PathSep:Char);
6568: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
6569: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
6570: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)');
6571: Function FileNameValidA( const FileName : AnsiString) : AnsiString');
6572: Function FileNameValid( const FileName : String) : String');
6573: Function FilePathA( const FileName,Path: AnsiString; const BasePath:AnsiString;const
      PathSep:Char):AnsiString;
6574: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String');
6575: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString');
6576: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String');
6577: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString');
6578: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString');
6579: Procedure CCopyFile( const FileName, DestName : String)');
6580: Procedure CMoveFile( const FileName, DestName : String)');
6581: Function CDeleteFiles( const FileMask : String) : Boolean');
6582: Function FileSeekEx( const FileHandle : TFileHandle; const FileOffset : Int64; const FilePosition :
      TFileSeekPosition) : Int64');
6583: Procedure FileCloseEx( const FileHandle : TFileHandle)');
6584: Function FileExistsA( const FileName : AnsiString) : Boolean');
6585:  Function CFileExists( const FileName : String) : Boolean');
6586:  Function CFileGetSize( const FileName : String) : Int64');
6587: Function FileGetDateTime( const FileName : String) : TDateTime');
6588: Function FileGetDateTime2( const FileName : String) : TDateTime');
6589: Function FileIsReadOnly( const FileName : String) : Boolean');
6590: Procedure FileDeleteEx( const FileName : String)');
```

```
6591: Procedure FileRenameEx( const OldFileName, NewFileName : String)');
6592: Function ReadFileStrA( const FileName : AnsiString; const FileSharing : TFileSharing; const
      FileCreationMode : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString');
6593: Function DirectoryEntryExists( const Name : String) : Boolean');
6594: Function DirectoryEntrySize( const Name : String) : Int64');
6595:  Function CDirectoryExists( const DirectoryName : String) : Boolean');
6596: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime');
6597: Procedure CDirectoryCreate( const DirectoryName : String)');
6598: Function GetFirstFileNameMatching( const FileMask : String) : String');
6599: Function DirEntryGetAttr( const FileName : AnsiString) : Integer');
6600: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean');
6601: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean');
6602:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
6603:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )');
6604: Function DriveIsValid( const Drive : Char) : Boolean');
6605: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType');
6606: Function DriveFreeSpace( const Path : AnsiString) : Int64');
6607:
6608: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
6609: begin
6610:  AddClassN(FindClass('TOBJECT'),'ETimers');
6611:  AddConstantN('TickFrequency','LongInt').SetInt( 1000);Function GetTick : LongWord');
6612: Function TickDelta( const D1, D2 : LongWord) : Integer');
6613: Function TickDeltaW( const D1, D2 : LongWord) : LongWord');
6614:   AddTypeS('THPTimer', 'Int64');
6615: Procedure StartTimer( var Timer : THPTimer)');
6616: Procedure StopTimer( var Timer : THPTimer)');
6617: Procedure ResumeTimer( var StoppedTimer : THPTimer)');
6618: Procedure InitStoppedTimer( var Timer : THPTimer)');
6619: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)');
6620: Function MillisecondsElapsed( const Timer : THPTimer; const TimerRunning : Boolean) : Integer');
6621: Function MicrosecondsElapsed( const Timer : THPTimer; const TimerRunning : Boolean) : Int64');
6622: Procedure WaitMicroseconds( const MicroSeconds : Integer)');
6623: Function GetHighPrecisionFrequency : Int64');
6624: Function GetHighPrecisionTimerOverhead : Int64');
6625: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)');
6626: Procedure SelfTestCTimer');
6627: end;
6628:
6629: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
6630: begin
6631:  Function RandomSeed : LongWord');
6632:  Procedure AddEntropy( const Value : LongWord)');
6633:  Function RandomUniform : LongWord;');
6634:  Function RandomUniform1( const N : Integer) : Integer;');
6635:  Function RandomBoolean : Boolean');
6636:  Function RandomByte : Byte');
6637:  Function RandomByteNonZero : Byte');
6638:  Function RandomWord : Word');
6639:  Function RandomInt64 : Int64;');
6640:  Function RandomInt641( const N : Int64) : Int64;');
6641:  Function RandomHex( const Digits : Integer) : String');
6642:  Function RandomFloat : Extended');
6643:  Function RandomAlphaStr( const Length : Integer) : AnsiString');
6644:  Function RandomPseudoWord( const Length : Integer) : AnsiString');
6645:  Function RandomPassword( const MinLength, MaxLength:Integer; const CaseSensitive,UseSymbols,UseNumbers:
      Boolean) : AnsiString');
6646:  Function mwcRandomLongWord : LongWord');
6647:  Function urnRandomLongWord : LongWord');
6648:  Function moaRandomFloat : Extended');
6649:  Function mwcRandomFloat : Extended');
6650:  Function RandomNormalF : Extended');
6651:  Procedure SelfTestCRandom');
6652: end;
6653:
6654:
6655: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
6656: begin
6657:  // CL.AddTypeS('PIntArray', '^TIntArray // will not work');
6658:  Addtypes('TConvertTabsProc' ,'function(const Line: AnsiString; TabWidth: integer): AnsiString');
6659:  Addtypes('TConvertTabsProcEx','function(const Line:AnsiString; TabWidth: integer;var HasTabs: boolean):
      AnsiString');
6660:  Function synMax( x, y : integer) : integer');
6661:  Function synMin( x, y : integer) : integer');
6662:  Function synMinMax( x, mi, ma : integer) : integer');
6663:  Procedure synSwapInt( var l, r : integer)');
6664:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint');
6665:  Function synMinPoint( const P1, P2 : TPoint) : TPoint');
6666:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray');
6667:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)');
6668:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc');
6669:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString');
6670:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx');
6671:  Function synConvertTabsEx( const Line : AnsiString; TabWidth : integer; var HasTabs : boolean) :
      AnsiString');
6672:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer');
```

```
6673:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer');
6674:  Function synCaretPos2CharIndex(Position,TabWidth:integer; const Line:string;var InsideTabChar: boolean):
      integer');
6675:  Function synStrScanForCharInSet( const Line : string; Start : integer; AChars : TSynIdentChars) :
      integer');
6676:  Function synStrRScanForCharInSet( const Line : string; Start : integer; AChars : TSynIdentChars) :
      integer');
6677:    CL.AddTypeS('TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
6678:     +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )');
6679:  CL.AddConstantN('C3_NONSPACING','LongInt').SetInt( 1);
6680:  CL.AddConstantN('C3_DIACRITIC','LongInt').SetInt( 2);
6681:  CL.AddConstantN('C3_VOWELMARK','LongInt').SetInt( 4);
6682:  CL.AddConstantN('C3_SYMBOL','LongInt').SetInt( 8);
6683:  CL.AddConstantN('C3_KATAKANA','LongWord').SetUInt( $0010);
6684:  CL.AddConstantN('C3_HIRAGANA','LongWord').SetUInt( $0020);
6685:  CL.AddConstantN('C3_HALFWIDTH','LongWord').SetUInt( $0040);
6686:  CL.AddConstantN('C3_FULLWIDTH','LongWord').SetUInt( $0080);
6687:  CL.AddConstantN('C3_IDEOGRAPH','LongWord').SetUInt( $0100);
6688:  CL.AddConstantN('C3_KASHIDA','LongWord').SetUInt( $0200);
6689:  CL.AddConstantN('C3_LEXICAL','LongWord').SetUInt( $0400);
6690:  CL.AddConstantN('C3_ALPHA','LongWord').SetUInt( $8000);
6691:  CL.AddConstantN('C3_NOTAPPLICABLE','LongInt').SetInt( 0);
6692:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer');
6693:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer');
6694:  Function synIsStringType( Value : Word) : TStringType');
6695:  Function synGetEOL( Line : PChar) : PChar');
6696:  Function synEncodeString( s : string) : string');
6697:  Function synDecodeString( s : string) : string');
6698:  Procedure synFreeAndNil( var Obj: TObject)');
6699:  Procedure synAssert( Expr : Boolean)');
6700:  Function synLastDelimiter( const Delimiters, S : string) : Integer');
6701:   CL.AddTypeS('TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )');
6702:   CL.AddTypeS('TReplaceFlags', 'set of TReplaceFlag )');
6703:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string');
6704:  Function synGetRValue( RGBValue : TColor) : byte');
6705:  Function synGetGValue( RGBValue : TColor) : byte');
6706:  Function synGetBValue( RGBValue : TColor) : byte');
6707:  Function synRGB( r, g, b : Byte) : Cardinal');
6708: //  CL.AddTypeS('THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
6709:    // +'lighter; Attri : TSynHighlighterAttributes; UniqueAttriName : string; Params  array of Pointer):
      Boolean');
6710:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean');
6711:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word');
6712:  Procedure synSynDrawGradient( const ACanvas : TCanvas; const AStartColor,AEndColor: TColor; ASteps:
      integer; const ARect : TRect; const AHorizontal : boolean)');
6713: end;
6714:
6715:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD');
6716:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD');
6717:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD');
6718:
6719:
6720: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
6721: begin
6722:  function CreateOleObject(const ClassName: String): IDispatch;');
6723:  function GetActiveOleObject(const ClassName: String): IDispatch;');
6724:  function ProgIDToClassID(const ProgID: string): TGUID;');
6725:  function ClassIDToProgID(const ClassID: TGUID): string;');
6726:  function CreateClassID: string;');
6727:  procedure OleError(ErrorCode: longint)');
6728:  procedure OleCheck(Result: HResult);');
6729: end;
6730:
6731:  Function xCreateOleObject( const ClassName : string) : Variant'); //or IDispatch
6732:  Function xGetActiveOleObject( const ClassName : string) : Variant');
6733:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult');
6734:  Function DllCanUnloadNow : HResult');
6735:  Function DllRegisterServer : HResult');
6736:  Function DllUnregisterServer : HResult');
6737:  Function VarFromInterface( Unknown : IUnknown) : Variant');
6738:  Function VarToInterface( const V : Variant) : IDispatch');
6739:  Function VarToAutoObject( const V : Variant) : TAutoObject');
6740:  //Procedure DispInvoke( Dispatch : IDispatch; CallDesc : PCallDesc; DispIDs : PDispIDList; Params :
      Pointer; Result : PVariant)');
6741:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)');
6742:  Procedure OleError( ErrorCode : HResult)');
6743:  Procedure OleCheck( Result : HResult)');
6744:  Function StringToClassID( const S : string) : TCLSID');
6745:  Function ClassIDToString( const ClassID : TCLSID) : string');
6746:  Function xProgIDToClassID( const ProgID : string) : TCLSID');
6747:  Function xClassIDToProgID( const ClassID : TCLSID) : string');
6748:  Function xWideCompareStr( const S1, S2 : WideString) : Integer');
6749:  Function xWideSameStr( const S1, S2 : WideString) : Boolean');
6750:  Function xGUIDToString( const ClassID : TGUID) : string');
6751:  Function xStringToGUID( const S : string) : TGUID');
```

```
6752:  Function xGetModuleName( Module : HMODULE) : string');
6753:  Function xAcquireExceptionObject : TObject');
6754:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer');
6755:  Function xUtf8Encode( const WS : WideString) : UTF8String');
6756:  Function xUtf8Decode( const S : UTF8String) : WideString');
6757:  Function xExcludeTrailingPathDelimiter( const S : string) : string');
6758:  Function xIncludeTrailingPathDelimiter( const S : string) : string');
6759:
6760:  Function XRTLHandleCOMException : HResult');
6761:  Procedure XRTLCheckArgument( Flag : Boolean)');
6762:  //Procedure XRTLCheckOutArgument( out Arg)');
6763:  Procedure XRTLInterfaceConnect(const Source:IUnknown; const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
6764:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)');
6765:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Integer):HResult
6766:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult');
6767:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult');
6768:  Procedure XRTLEnumActiveObjects( Strings : TStrings)');
6769:
6770: function   XRTLDefaultCategoryManager: IUnknown;
6771: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
6772: // ICatRegister helper functions
6773: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
6774:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
6775:                                        const CategoryManager: IUnknown = nil): HResult;
6776: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
6777:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
6778:                                        const CategoryManager: IUnknown = nil): HResult;
6779: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
6780:                                        const CategoryManager: IUnknown = nil): HResult;
6781: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
6782:                                          const CategoryManager: IUnknown = nil): HResult;
6783: // ICatInformation helper functions
6784: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
6785:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
6786:                                       const CategoryManager: IUnknown = nil): HResult;
6787: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
6788:                                const CategoryManager: IUnknown = nil): HResult;
6789: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
6790:                                     const CategoryManager: IUnknown = nil): HResult;
6791: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
6792:                                      const CategoryManager: IUnknown = nil): HResult;
6793: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
6794:                    const ADelete: Boolean = True): WideString;
6795: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
6796: Function XRTLGetVariantAsString( const Value : Variant) : string');
6797: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime');
6798: Function XRTLGetTimeZones : TXRTLTimeZones');
6799: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime');
6800: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime');
6801: Function GMTNow : TDateTime');
6802: Function GMTToLocalTime( GMT : TDateTime) : TDateTime');
6803: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime');
6804: Procedure XRTLNotImplemented');
6805: Procedure XRTLRaiseError( E : Exception)');
6806: Procedure XRTLInvalidOperation( ClassName : string; OperationName : string; Description : string)');
6807:
6808:
6809: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
6810: begin
6811:   SIRegister_IXRTLValue(CL);
6812:   SIRegister_TXRTLValue(CL);
6813:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work');
6814:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue');
6815: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;');
6816: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;');
6817: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal');
6818: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal');
6819: Function XRTLValue1( const AValue : Integer) : IXRTLValue;');
6820: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;');
6821: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer');
6822: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer');
6823: Function XRTLValue2( const AValue : Int64) : IXRTLValue;');
6824: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;');
6825: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64');
6826: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64');
6827: Function XRTLValue3( const AValue : Single) : IXRTLValue;');
6828: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;');
6829: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single');
6830: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single');
6831: Function XRTLValue4( const AValue : Double) : IXRTLValue;');
6832: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;');
6833: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double');
6834: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double');
6835: Function XRTLValue5( const AValue : Extended) : IXRTLValue;');
```

```
6836: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;');
6837: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended');
6838: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended');
6839: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;');
6840: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;');
6841: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;');
6842:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;');
6843: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;');
6844: Function XRTLValue7( const AValue : WideString) : IXRTLValue;');
6845: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;');
6846: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString');
6847: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString');
6848: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;');
6849: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;');
6850: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;');
6851: Function XRTLGetAsObjectDef(const IValue : IXRTLValue; const DefValue:TObject; const
      ADetachOwnership:Boolean) : TObject;');
6852: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;');
6853: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;');
6854: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer');
6855: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer');
6856: Function XRTLValueV( const AValue : Variant) : IXRTLValue;');
6857: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;');
6858: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant');
6859: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant');
6860: Function XRTLValue10( const AValue : Currency) : IXRTLValue;');
6861: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;');
6862: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency');
6863: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency');
6864: Function XRTLValue11( const AValue : Comp) : IXRTLValue;');
6865: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;');
6866: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp');
6867: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp');
6868: Function XRTLValue12( const AValue : TClass) : IXRTLValue;');
6869: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;');
6870: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass');
6871: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass');
6872: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;');
6873: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;');
6874: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID');
6875: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID');
6876: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;');
6877: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;');
6878: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean');
6879: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean');
6880: end;
6881:
6882: //*****************************unit uPSI_GR32;****************************************
6883:
6884:  Function Color32( WinColor : TColor) : TColor32;');
6885:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;');
6886:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;');
6887:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32');
6888:  Function WinColor( Color32 : TColor32) : TColor');
6889:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32');
6890:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)');
6891:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)');
6892:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components');
6893:  Function RedComponent( Color32 : TColor32) : Integer');
6894:  Function GreenComponent( Color32 : TColor32) : Integer');
6895:  Function BlueComponent( Color32 : TColor32) : Integer');
6896:  Function AlphaComponent( Color32 : TColor32) : Integer');
6897:  Function Intensity( Color32 : TColor32) : Integer');
6898:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32');
6899:  Function HSLtoRGB( H, S, L : Single) : TColor32;');
6900:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);');
6901:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;');
6902:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);');
6903:  Function WinPalette( const P : TPalette32) : HPALETTE');
6904:  Function FloatPoint( X, Y : Single) : TFloatPoint;');
6905:  Function FloatPoint1( const P : TPoint) : TFloatPoint;');
6906:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;');
6907:  Function FixedPoint( X, Y : Integer) : TFixedPoint;');
6908:  Function FixedPoint1( X, Y : Single) : TFixedPoint;');
6909:  Function FixedPoint2( const P : TPoint) : TFixedPoint;');
6910:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;');
6911:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )');
6912:  Function MakeRect( const L, T, R, B : Integer) : TRect;');
6913:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;');
6914:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;');
6915:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;');
6916:  Function FixedRect1( const ARect : TRect) : TRect;');
6917:  Function FixedRect2( const FR : TFloatRect) : TRect;');
6918:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;');
6919:  Function FloatRect1( const ARect : TRect) : TFloatRect;');
6920:  Function FloatRect2( const FXR : TRect) : TFloatRect;');
```

```
6921:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;');
6922:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;');
6923:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;');
6924:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;');
6925:  Function GEqualRect( const R1, R2 : TRect) : Boolean;');
6926:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;');
6927:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);');
6928:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);');
6929:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);');
6930:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);');
6931:  Function IsRectEmpty( const R : TRect) : Boolean;');
6932:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;');
6933:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;');
6934:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;');
6935:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;');
6936:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;');
6937:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;');
6938:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;');
6939:  Function MessageBeep( uType : UINT) : BOOL');
6940:  Function ShowCursor( bShow : BOOL) : Integer');
6941:  Function SetCursorPos( X, Y : Integer) : BOOL');
6942:  Function SetCursor( hCursor : HICON) : HCURSOR');
6943:  Function GetCursorPos( var lpPoint : TPoint) : BOOL');
6944:  //Function ClipCursor( lpRect : PRect) : BOOL');
6945:  Function GetClipCursor( var lpRect : TRect) : BOOL');
6946:  Function GetCursor : HCURSOR');
6947:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL');
6948:  Function GetCaretBlinkTime : UINT');
6949:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL');
6950:  Function DestroyCaret : BOOL');
6951:  Function HideCaret( hWnd : HWND) : BOOL');
6952:  Function ShowCaret( hWnd : HWND) : BOOL');
6953:  Function SetCaretPos( X, Y : Integer) : BOOL');
6954:  Function GetCaretPos( var lpPoint : TPoint) : BOOL');
6955:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL');
6956:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL');
6957:  Function MapWindowPoints( hWndFrom, hWndTo : HWND; var lpPoints, cPoints : UINT) : Integer');
6958:  Function WindowFromPoint( Point : TPoint) : HWND');
6959:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND');
6960:
6961:
6962: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
6963: begin
6964:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError');
6965:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
6966:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;');
6967:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;');
6968:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean');
6969:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)');
6970:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)');
6971:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)');
6972:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
6973:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
6974:    //+'tedRangeBuffer; MoreData : Boolean; end');
6975:  Function NtfsSetSparse( const FileName : string) : Boolean');
6976:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean');
6977:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean');
6978:  //Function NtfsQueryAllocRanges(const FileName: string;Offset,Count:Int64; var Ranges:TNtfsAllocRanges):
     Boolean');
6979:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges; Index : Integer) :
     TFileAllocatedRangeBuffer');
6980:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean');
6981:  Function NtfsGetSparse( const FileName : string) : Boolean');
6982:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean');
6983:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean');
6984:  //Function NtfsGetReparsePoint( const FileName : string; var ReparseData : TReparseGuidDataBuffer) :
     Boolean');
6985:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean');
6986:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean');
6987:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean');
6988:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean');
6989:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean');
6990:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean');
6991:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
6992:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean');
6993:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean');
6994:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean');
6995:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean');
6996:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean');
6997:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean');
6998:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean');
6999:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean');
7000:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
7001:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )');
7002:   AddTypeS('TStreamIds', 'set of TStreamId');
7003:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
```

```
7004:    +': ___Pointer; StreamIds : TStreamIds; end');
7005:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
7006:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end');
7007:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
7008:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean');
7009:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean');
7010:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean');
7011:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
7012:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean');
7013:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow: Cardinal;const
      List:TStrings):Boolean;
7014:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean');
7015:
7016:  Function JclAppInstances : TJclAppInstances;');
7017:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;');
7018:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) :
      TJclAppInstDataKind');
7019:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)');
7020:  Procedure ReadMessageString( const Message : TMessage; var S : string)');
7021:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)');
7022:
7023: //*************************unit uPSI_mORMotReport;
7024: Procedure SetCurrentPrinterAsDefault');
7025: Function CurrentPrinterName : string');
7026: Function mCurrentPrinterPaperSize : string');
7027: Procedure UseDefaultPrinter');
7028:
7029: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
7030: begin
7031:   with Cl.AddClassN(cl.FindClass('TOBJECT'), 'TStream') do begin
7032:     IsAbstract := True;
7033:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint');
7034:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint');
7035:     function Read(Buffer:String;Count:LongInt):LongInt');
7036:     function Write(Buffer:String;Count:LongInt):LongInt');
7037:
7038:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)');
7039:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)');
7040:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)');
7041:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)');
7042:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)');
7043:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)');
7044:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)');
7045:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)');
7046:
7047:     function Seek(Offset:LongInt;Origin:Word):LongInt');
7048:     procedure ReadBuffer(Buffer:String;Count:LongInt)');
7049:     procedure WriteBuffer(Buffer:String;Count:LongInt)');
7050:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)');
7051:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)');
7052:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)');
7053:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)');
7054:
7055:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)');
7056:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)');
7057:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)');
7058:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)');
7059:
7060:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)');
7061:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)');
7062:
7063:      function InstanceSize: Longint');
7064:     Procedure FixupResourceHeader( FixupInfo : Integer)');
7065:     Procedure ReadResHeader');
7066:
7067:     {$IFDEF DELPHI4UP}
7068:     function CopyFrom(Source:TStream;Count:Int64):LongInt');
7069:     {$ELSE}
7070:     function CopyFrom(Source:TStream;Count:Integer):LongInt');
7071:     {$ENDIF}
7072:     RegisterProperty('Position', 'LongInt', iptrw);
7073:     RegisterProperty('Size', 'LongInt', iptrw);
7074:   end;
7075: end;
7076:
7077:
7078:
7079: { ***************************************************************
7080:   Unit DMATH - Interface for DMATH.DLL
7081:    ************************************************************** }
7082: // see more docs/dmath_manual.pdf
7083:
7084: Function InitEval : Integer');
7085: Procedure SetVariable( VarName : Char; Value : Float)');
7086: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)');
7087: Function Eval( ExpressionString : String) : Float');
```

```
7088:
7089: unit dmath; //types are in built, others are external in DLL
7090: interface
7091: {$IFDEF DELPHI}
7092: uses
7093:   StdCtrls, Graphics;
7094: {$ENDIF}
7095: { ---------------------------------------------------------------
7096:   Types and constants
7097:   --------------------------------------------------------------- }
7098: {$i types.inc}
7099: { ---------------------------------------------------------------
7100:   Error handling
7101:   --------------------------------------------------------------- }
7102: procedure SetErrCode(ErrCode : Integer); external 'dmath';
7103: { Sets the error code }
7104: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
7105: { Sets error code and default function value }
7106: function MathErr : Integer; external 'dmath';
7107: { Returns the error code }
7108: { ---------------------------------------------------------------
7109:   Dynamic arrays
7110:   --------------------------------------------------------------- }
7111: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
7112: { Sets the auto-initialization of arrays }
7113: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
7114: { Creates floating point vector V[0..Ub] }
7115: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
7116: { Creates integer vector V[0..Ub] }
7117: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
7118: { Creates complex vector V[0..Ub] }
7119: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
7120: { Creates boolean vector V[0..Ub] }
7121: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
7122: { Creates string vector V[0..Ub] }
7123: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
7124: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
7125: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
7126: { Creates integer matrix A[0..Ub1, 0..Ub2] }
7127: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
7128: { Creates complex matrix A[0..Ub1, 0..Ub2] }
7129: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
7130: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
7131: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
7132: { Creates string matrix A[0..Ub1, 0..Ub2] }
7133: { ---------------------------------------------------------------
7134:   Minimum, maximum, sign and exchange
7135:   --------------------------------------------------------------- }
7136: function FMin(X, Y : Float) : Float; external 'dmath';
7137: { Minimum of 2 reals }
7138: function FMax(X, Y : Float) : Float; external 'dmath';
7139: { Maximum of 2 reals }
7140: function IMin(X, Y : Integer) : Integer; external 'dmath';
7141: { Minimum of 2 integers }
7142: function IMax(X, Y : Integer) : Integer; external 'dmath';
7143: { Maximum of 2 integers }
7144: function Sgn(X : Float) : Integer; external 'dmath';
7145: { Sign (returns 1 if X = 0) }
7146: function Sgn0(X : Float) : Integer; external 'dmath';
7147: { Sign (returns 0 if X = 0) }
7148: function DSgn(A, B : Float) : Float; external 'dmath';
7149: { Sgn(B) * |A| }
7150: procedure FSwap(var X, Y : Float); external 'dmath';
7151: { Exchange 2 reals }
7152: procedure ISwap(var X, Y : Integer); external 'dmath';
7153: { Exchange 2 integers }
7154: { ---------------------------------------------------------------
7155:   Rounding functions
7156:   --------------------------------------------------------------- }
7157: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
7158: { Rounds X to N decimal places }
7159: function Ceil(X : Float) : Integer; external 'dmath';
7160: { Ceiling function }
7161: function Floor(X : Float) : Integer; external 'dmath';
7162: { Floor function }
7163: { ---------------------------------------------------------------
7164:   Logarithms, exponentials and power
7165:   --------------------------------------------------------------- }
7166: function Expo(X : Float) : Float; external 'dmath';
7167: { Exponential }
7168: function Exp2(X : Float) : Float; external 'dmath';
7169: { 2^X }
7170: function Exp10(X : Float) : Float; external 'dmath';
7171: { 10^X }
7172: function Log(X : Float) : Float; external 'dmath';
7173: { Natural log }
```

```
7174: function Log2(X : Float) : Float; external 'dmath';
7175: { Log, base 2 }
7176: function Log10(X : Float) : Float; external 'dmath';
7177: { Decimal log }
7178: function LogA(X, A : Float) : Float; external 'dmath';
7179: { Log, base A }
7180: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
7181: { X^N }
7182: function Power(X, Y : Float) : Float; external 'dmath';
7183: { X^Y, X >= 0 }
7184: { ------------------------------------------------------------------
7185:   Trigonometric functions
7186:   ------------------------------------------------------------------ }
7187: function Pythag(X, Y : Float) : Float; external 'dmath';
7188: { Sqrt(X^2 + Y^2) }
7189: function FixAngle(Theta : Float) : Float; external 'dmath';
7190: { Set Theta in -Pi..Pi }
7191: function Tan(X : Float) : Float; external 'dmath';
7192: { Tangent }
7193: function ArcSin(X : Float) : Float; external 'dmath';
7194: { Arc sinus }
7195: function ArcCos(X : Float) : Float; external 'dmath';
7196: { Arc cosinus }
7197: function ArcTan2(Y, X : Float) : Float; external 'dmath';
7198: { Angle (Ox, OM) with M(X,Y) }
7199: { ------------------------------------------------------------------
7200:   Hyperbolic functions
7201:   ------------------------------------------------------------------ }
7202: function Sinh(X : Float) : Float; external 'dmath';
7203: { Hyperbolic sine }
7204: function Cosh(X : Float) : Float; external 'dmath';
7205: { Hyperbolic cosine }
7206: function Tanh(X : Float) : Float; external 'dmath';
7207: { Hyperbolic tangent }
7208: function ArcSinh(X : Float) : Float; external 'dmath';
7209: { Inverse hyperbolic sine }
7210: function ArcCosh(X : Float) : Float; external 'dmath';
7211: { Inverse hyperbolic cosine }
7212: function ArcTanh(X : Float) : Float; external 'dmath';
7213: { Inverse hyperbolic tangent }
7214: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
7215: { Sinh & Cosh }
7216: { ------------------------------------------------------------------
7217:   Gamma function and related functions
7218:   ------------------------------------------------------------------ }
7219: function Fact(N : Integer) : Float; external 'dmath';
7220: { Factorial }
7221: function SgnGamma(X : Float) : Integer; external 'dmath';
7222: { Sign of Gamma function }
7223: function Gamma(X : Float) : Float; external 'dmath';
7224: { Gamma function }
7225: function LnGamma(X : Float) : Float; external 'dmath';
7226: { Logarithm of Gamma function }
7227: function Stirling(X : Float) : Float; external 'dmath';
7228: { Stirling's formula for the Gamma function }
7229: function StirLog(X : Float) : Float; external 'dmath';
7230: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
7231: function DiGamma(X : Float ) : Float; external 'dmath';
7232: { Digamma function }
7233: function TriGamma(X : Float ) : Float; external 'dmath';
7234: { Trigamma function }
7235: function IGamma(A, X : Float) : Float; external 'dmath';
7236: { Incomplete Gamma function}
7237: function JGamma(A, X : Float) : Float; external 'dmath';
7238: { Complement of incomplete Gamma function }
7239: function InvGamma(A, P : Float) : Float; external 'dmath';
7240: { Inverse of incomplete Gamma function }
7241: function Erf(X : Float) : Float; external 'dmath';
7242: { Error function }
7243: function Erfc(X : Float) : Float; external 'dmath';
7244: { Complement of error function }
7245: { ------------------------------------------------------------------
7246:   Beta function and related functions
7247:   ------------------------------------------------------------------ }
7248: function Beta(X, Y : Float) : Float; external 'dmath';
7249: { Beta function }
7250: function IBeta(A, B, X : Float) : Float; external 'dmath';
7251: { Incomplete Beta function }
7252: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
7253: { Inverse of incomplete Beta function }
7254: { ------------------------------------------------------------------
7255:   Lambert's function
7256:   ------------------------------------------------------------------ }
7257: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
7258:  ------------------------------------------------------------------
7259:   Binomial distribution
```

```
7260:   ------------------------------------------------------------- }
7261: function Binomial(N, K : Integer) : Float; external 'dmath';
7262: { Binomial coefficient C(N,K) }
7263: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
7264: { Probability of binomial distribution }
7265: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
7266: { Cumulative probability for binomial distrib. }
7267: { -------------------------------------------------------------
7268:   Poisson distribution
7269:   ------------------------------------------------------------- }
7270: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
7271: { Probability of Poisson distribution }
7272: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
7273: { Cumulative probability for Poisson distrib. }
7274: { -------------------------------------------------------------
7275:   Exponential distribution
7276:   ------------------------------------------------------------- }
7277: function DExpo(A, X : Float) : Float; external 'dmath';
7278: { Density of exponential distribution with parameter A }
7279: function FExpo(A, X : Float) : Float; external 'dmath';
7280: { Cumulative probability function for exponential dist. with parameter A }
7281: { -------------------------------------------------------------
7282:   Standard normal distribution
7283:   ------------------------------------------------------------- }
7284: function DNorm(X : Float) : Float; external 'dmath';
7285: { Density of standard normal distribution }
7286: function FNorm(X : Float) : Float; external 'dmath';
7287: { Cumulative probability for standard normal distrib. }
7288: function PNorm(X : Float) : Float; external 'dmath';
7289: { Prob(|U| > X) for standard normal distrib. }
7290: function InvNorm(P : Float) : Float; external 'dmath';
7291: { Inverse of standard normal distribution }
7292: { -------------------------------------------------------------
7293:   Student's distribution
7294:   ------------------------------------------------------------- }
7295: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
7296: { Density of Student distribution with Nu d.o.f. }
7297: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
7298: { Cumulative probability for Student distrib. with Nu d.o.f. }
7299: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
7300: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
7301: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
7302: { Inverse of Student's t-distribution function }
7303: { -------------------------------------------------------------
7304:   Khi-2 distribution
7305:   ------------------------------------------------------------- }
7306: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
7307: { Density of Khi-2 distribution with Nu d.o.f. }
7308: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
7309: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
7310: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
7311: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
7312: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
7313: { Inverse of Khi-2 distribution function }
7314: { -------------------------------------------------------------
7315:   Fisher-Snedecor distribution
7316:   ------------------------------------------------------------- }
7317: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
7318: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7319: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
7320: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
7321: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
7322: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
7323: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
7324: { Inverse of Snedecor's F-distribution function }
7325: { -------------------------------------------------------------
7326:   Beta distribution
7327:   ------------------------------------------------------------- }
7328: function DBeta(A, B, X : Float) : Float; external 'dmath';
7329: { Density of Beta distribution with parameters A and B }
7330: function FBeta(A, B, X : Float) : Float; external 'dmath';
7331: { Cumulative probability for Beta distrib. with param. A and B }
7332: { -------------------------------------------------------------
7333:   Gamma distribution
7334:   ------------------------------------------------------------- }
7335: function DGamma(A, B, X : Float) : Float; external 'dmath';
7336: { Density of Gamma distribution with parameters A and B }
7337: function FGamma(A, B, X : Float) : Float; external 'dmath';
7338: { Cumulative probability for Gamma distrib. with param. A and B }
7339: { -------------------------------------------------------------
7340:   Expression evaluation
7341:   ------------------------------------------------------------- }
7342: function InitEval : Integer; external 'dmath';
7343: { Initializes built-in functions and returns their number }
7344: function Eval(ExpressionString : String) : Float; external 'dmath';
7345: { Evaluates an expression at run-time }
```

```
7346: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
7347: { Assigns a value to a variable }
7348: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
7349: { Adds a function to the parser }
7350: { ---------------------------------------------------------------
7351:   Matrices and linear equations
7352:   --------------------------------------------------------------- }
7353: procedure GaussJordan(A           : TMatrix;
7354:                       Lb, Ub1, Ub2 : Integer;
7355:                       var Det      : Float); external 'dmath';
7356: { Transforms a matrix according to the Gauss-Jordan method }
7357: procedure LinEq(A       : TMatrix;
7358:                 B       : TVector;
7359:                 Lb, Ub  : Integer;
7360:                 var Det : Float); external 'dmath';
7361: { Solves a linear system according to the Gauss-Jordan method }
7362: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
7363: { Cholesky factorization of a positive definite symmetric matrix }
7364: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
7365: { LU decomposition }
7366: procedure LU_Solve(A      : TMatrix;
7367:                    B      : TVector;
7368:                    Lb, Ub : Integer;
7369:                    X      : TVector); external 'dmath';
7370: { Solution of linear system from LU decomposition }
7371: procedure QR_Decomp(A            : TMatrix;
7372:                     Lb, Ub1, Ub2 : Integer;
7373:                     R            : TMatrix); external 'dmath';
7374: { QR decomposition }
7375: procedure QR_Solve(Q, R         : TMatrix;
7376:                    B            : TVector;
7377:                    Lb, Ub1, Ub2 : Integer;
7378:                    X            : TVector); external 'dmath';
7379: { Solution of linear system from QR decomposition }
7380: procedure SV_Decomp(A            : TMatrix;
7381:                     Lb, Ub1, Ub2 : Integer;
7382:                     S            : TVector;
7383:                     V            : TMatrix); external 'dmath';
7384: { Singular value decomposition }
7385: procedure SV_SetZero(S      : TVector;
7386:                      Lb, Ub : Integer;
7387:                      Tol    : Float); external 'dmath';
7388: { Set lowest singular values to zero }
7389: procedure SV_Solve(U            : TMatrix;
7390:                    S            : TVector;
7391:                    V            : TMatrix;
7392:                    B            : TVector;
7393:                    Lb, Ub1, Ub2 : Integer;
7394:                    X            : TVector); external 'dmath';
7395: { Solution of linear system from SVD }
7396: procedure SV_Approx(U            : TMatrix;
7397:                     S            : TVector;
7398:                     V            : TMatrix;
7399:                     Lb, Ub1, Ub2 : Integer;
7400:                     A            : TMatrix); external 'dmath';
7401: { Matrix approximation from SVD }
7402: procedure EigenVals(A      : TMatrix;
7403:                     Lb, Ub : Integer;
7404:                     Lambda : TCompVector); external 'dmath';
7405: { Eigenvalues of a general square matrix }
7406: procedure EigenVect(A      : TMatrix;
7407:                     Lb, Ub : Integer;
7408:                     Lambda : TCompVector;
7409:                     V      : TMatrix); external 'dmath';
7410: { Eigenvalues and eigenvectors of a general square matrix }
7411: procedure EigenSym(A      : TMatrix;
7412:                    Lb, Ub : Integer;
7413:                    Lambda : TVector;
7414:                    V      : TMatrix); external 'dmath';
7415: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
7416: procedure Jacobi(A               : TMatrix;
7417:                  Lb, Ub, MaxIter : Integer;
7418:                  Tol             : Float;
7419:                  Lambda          : TVector;
7420:                  V               : TMatrix); external 'dmath';
7421: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
7422: { ---------------------------------------------------------------
7423:   Optimization
7424:   --------------------------------------------------------------- }
7425: procedure MinBrack(Func                    : TFunc;
7426:                    var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
7427: { Brackets a minimum of a function }
7428: procedure GoldSearch(Func          : TFunc;
7429:                      A, B          : Float;
7430:                      MaxIter       : Integer;
7431:                      Tol           : Float;
```

```
7432:                     var Xmin, Ymin : Float); external 'dmath';
7433: { Minimization of a function of one variable (golden search) }
7434: procedure LinMin(Func      : TFuncNVar;
7435:                  X, DeltaX : TVector;
7436:                  Lb, Ub    : Integer;
7437:                  var R     : Float;
7438:                  MaxIter   : Integer;
7439:                  Tol       : Float;
7440:                  var F_min : Float); external 'dmath';
7441: { Minimization of a function of several variables along a line }
7442: procedure Newton(Func      : TFuncNVar;
7443:                  HessGrad  : THessGrad;
7444:                  X         : TVector;
7445:                  Lb, Ub    : Integer;
7446:                  MaxIter   : Integer;
7447:                  Tol       : Float;
7448:                  var F_min : Float;
7449:                  G         : TVector;
7450:                  H_inv     : TMatrix;
7451:                  var Det   : Float); external 'dmath';
7452: { Minimization of a function of several variables (Newton's method) }
7453: procedure SaveNewton(FileName : string); external 'dmath';
7454: { Save Newton iterations in a file }
7455: procedure Marquardt(Func      : TFuncNVar;
7456:                     HessGrad  : THessGrad;
7457:                     X         : TVector;
7458:                     Lb, Ub    : Integer;
7459:                     MaxIter   : Integer;
7460:                     Tol       : Float;
7461:                     var F_min : Float;
7462:                     G         : TVector;
7463:                     H_inv     : TMatrix;
7464:                     var Det   : Float); external 'dmath';
7465: { Minimization of a function of several variables (Marquardt's method) }
7466: procedure SaveMarquardt(FileName : string); external 'dmath';
7467: { Save Marquardt iterations in a file }
7468: procedure BFGS(Func      : TFuncNVar;
7469:                Gradient  : TGradient;
7470:                X         : TVector;
7471:                Lb, Ub    : Integer;
7472:                MaxIter   : Integer;
7473:                Tol       : Float;
7474:                var F_min : Float;
7475:                G         : TVector;
7476:                H_inv     : TMatrix); external 'dmath';
7477: { Minimization of a function of several variables (BFGS method) }
7478: procedure SaveBFGS(FileName : string); external 'dmath';
7479: { Save BFGS iterations in a file }
7480: procedure Simplex(Func      : TFuncNVar;
7481:                   X         : TVector;
7482:                   Lb, Ub    : Integer;
7483:                   MaxIter   : Integer;
7484:                   Tol       : Float;
7485:                   var F_min : Float); external 'dmath';
7486: { Minimization of a function of several variables (Simplex) }
7487: procedure SaveSimplex(FileName : string); external 'dmath';
7488: { Save Simplex iterations in a file }
7489: { ------------------------------------------------------------------
7490:   Nonlinear equations
7491:   ------------------------------------------------------------------ }
7492: procedure RootBrack(Func             : TFunc;
7493:                     var X, Y, FX, FY : Float); external 'dmath';
7494: { Brackets a root of function Func between X and Y }
7495: procedure Bisect(Func     : TFunc;
7496:                  var X, Y : Float;
7497:                  MaxIter  : Integer;
7498:                  Tol      : Float;
7499:                  var F    : Float); external 'dmath';
7500: { Bisection method }
7501: procedure Secant(Func     : TFunc;
7502:                  var X, Y : Float;
7503:                  MaxIter  : Integer;
7504:                  Tol      : Float;
7505:                  var F    : Float); external 'dmath';
7506: { Secant method }
7507: procedure NewtEq(Func, Deriv : TFunc;
7508:                  var X       : Float;
7509:                  MaxIter     : Integer;
7510:                  Tol         : Float;
7511:                  var F       : Float); external 'dmath';
7512: { Newton-Raphson method for a single nonlinear equation }
7513: procedure NewtEqs(Equations : TEquations;
7514:                   Jacobian  : TJacobian;
7515:                   X, F      : TVector;
7516:                   Lb, Ub    : Integer;
7517:                   MaxIter   : Integer;
```

```
7518:                  Tol       : Float); external 'dmath';
7519: { Newton-Raphson method for a system of nonlinear equations }
7520: procedure Broyden(Equations : TEquations;
7521:                   X, F      : TVector;
7522:                   Lb, Ub    : Integer;
7523:                   MaxIter   : Integer;
7524:                   Tol       : Float); external 'dmath';
7525: { Broyden's method for a system of nonlinear equations }
7526: { ---------------------------------------------------------------
7527:   Polynomials and rational fractions
7528:   --------------------------------------------------------------- }
7529: function Poly(X    : Float;
7530:               Coef : TVector;
7531:               Deg  : Integer) : Float; external 'dmath';
7532: { Evaluates a polynomial }
7533: function RFrac(X          : Float;
7534:                Coef       : TVector;
7535:                Deg1, Deg2 : Integer) : Float; external 'dmath';
7536: { Evaluates a rational fraction }
7537: function RootPol1(A, B  : Float;
7538:                   var X : Float) : Integer; external 'dmath';
7539: { Solves the linear equation A + B * X = 0 }
7540: function RootPol2(Coef : TVector;
7541:                   Z    : TCompVector) : Integer; external 'dmath';
7542: { Solves a quadratic equation }
7543: function RootPol3(Coef : TVector;
7544:                   Z    : TCompVector) : Integer; external 'dmath';
7545: { Solves a cubic equation }
7546: function RootPol4(Coef : TVector;
7547:                   Z    : TCompVector) : Integer; external 'dmath';
7548: { Solves a quartic equation }
7549: function RootPol(Coef : TVector;
7550:                  Deg  : Integer;
7551:                  Z    : TCompVector) : Integer; external 'dmath';
7552: { Solves a polynomial equation }
7553: function SetRealRoots(Deg : Integer;
7554:                       Z   : TCompVector;
7555:                       Tol : Float) : Integer; external 'dmath';
7556: { Set the imaginary part of a root to zero }
7557: procedure SortRoots(Deg : Integer;
7558:                     Z   : TCompVector); external 'dmath';
7559: { Sorts the roots of a polynomial }
7560: { ---------------------------------------------------------------
7561:   Numerical integration and differential equations
7562:   --------------------------------------------------------------- }
7563: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
7564: { Integration by trapezoidal rule }
7565: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
7566: { Integral from A to B }
7567: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
7568: { Integral from 0 to B }
7569: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
7570: { Convolution product at time T }
7571: procedure ConvTrap(Func1, Func2 : TFunc; T, Y : TVector; N : Integer); external 'dmath';
7572: { Convolution by trapezoidal rule }
7573: procedure RKF45(F                   : TDiffEqs;
7574:                 Neqn                : Integer;
7575:                 Y, Yp               : TVector;
7576:                 var T               : Float;
7577:                 Tout, RelErr, AbsErr : Float;
7578:                 var Flag            : Integer); external 'dmath';
7579: { Integration of a system of differential equations }
7580: { ---------------------------------------------------------------
7581:   Fast Fourier Transform
7582:   --------------------------------------------------------------- }
7583: procedure FFT(NumSamples        : Integer;
7584:               InArray, OutArray : TCompVector); external 'dmath';
7585: { Fast Fourier Transform }
7586: procedure IFFT(NumSamples        : Integer;
7587:                InArray, OutArray : TCompVector); external 'dmath';
7588: { Inverse Fast Fourier Transform }
7589: procedure FFT_Integer(NumSamples     : Integer;
7590:                       RealIn, ImagIn : TIntVector;
7591:                       OutArray       : TCompVector); external 'dmath';
7592: { Fast Fourier Transform for integer data }
7593: procedure FFT_Integer_Cleanup; external 'dmath';
7594: { Clear memory after a call to FFT_Integer }
7595: procedure CalcFrequency(NumSamples,
7596:                         FrequencyIndex : Integer;
7597:                         InArray        : TCompVector;
7598:                         var FFT        : Complex); external 'dmath';
7599: { Direct computation of Fourier transform }
7600: { ---------------------------------------------------------------
7601:   Random numbers
7602:   --------------------------------------------------------------- }
7603: procedure SetRNG(RNG : RNG_Type); external 'dmath';
```

```
7604: { Select generator }
7605: procedure InitGen(Seed : RNG_IntType); external 'dmath';
7606: { Initialize generator }
7607: function IRanGen : RNG_IntType; external 'dmath';
7608: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
7609: function IRanGen31 : RNG_IntType; external 'dmath';
7610: { 31-bit random integer in [0 .. 2^31 - 1] }
7611: function RanGen1 : Float; external 'dmath';
7612: { 32-bit random real in [0,1] }
7613: function RanGen2 : Float; external 'dmath';
7614: { 32-bit random real in [0,1) }
7615: function RanGen3 : Float; external 'dmath';
7616: { 32-bit random real in (0,1) }
7617: function RanGen53 : Float; external 'dmath';
7618: { 53-bit random real in [0,1) }
7619: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
7620: { Initializes the 'Multiply with carry' random number generator }
7621: function IRanMWC : RNG_IntType; external 'dmath';
7622: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
7623: procedure InitMT(Seed : RNG_IntType); external 'dmath';
7624: { Initializes Mersenne Twister generator with a seed }
7625: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
7626:                         KeyLength : Word); external 'dmath';
7627: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
7628: function IRanMT : RNG_IntType; external 'dmath';
7629: { Random integer from MT generator }
7630: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
7631: { Initializes the UVAG generator with a string }
7632: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
7633: { Initializes the UVAG generator with an integer }
7634: function IRanUVAG : RNG_IntType; external 'dmath';
7635: { Random integer from UVAG generator }
7636: function RanGaussStd : Float; external 'dmath';
7637: { Random number from standard normal distribution }
7638: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
7639: { Random number from normal distrib. with mean Mu and S. D. Sigma }
7640: procedure RanMult(M      : TVector; L      : TMatrix;
7641:                   Lb, Ub : Integer;
7642:                   X      : TVector); external 'dmath';
7643: { Random vector from multinormal distribution (correlated) }
7644: procedure RanMultIndep(M, S   : TVector;
7645:                        Lb, Ub : Integer;
7646:                        X      : TVector); external 'dmath';
7647: { Random vector from multinormal distribution (uncorrelated) }
7648: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
7649: { Initializes Metropolis-Hastings parameters }
7650: procedure GetMHParams(var NCycles, MaxSim, SavedSim : Integer); external 'dmath';
7651: { Returns Metropolis-Hastings parameters }
7652: procedure Hastings(Func      : TFuncNVar;
7653:                    T         : Float;
7654:                    X         : TVector;
7655:                    V         : TMatrix;
7656:                    Lb, Ub    : Integer;
7657:                    Xmat      : TMatrix;
7658:                    X_min     : TVector;
7659:                    var F_min : Float); external 'dmath';
7660: { Simulation of a probability density function by Metropolis-Hastings }
7661: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
7662: { Initializes Simulated Annealing parameters }
7663: procedure SA_CreateLogFile(FileName : String); external 'dmath';
7664: { Initializes log file }
7665: procedure SimAnn(Func          : TFuncNVar;
7666:                  X, Xmin, Xmax : TVector;
7667:                  Lb, Ub        : Integer;
7668:                  var F_min     : Float); external 'dmath';
7669: { Minimization of a function of several var. by simulated annealing }
7670: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
7671: { Initializes Genetic Algorithm parameters }
7672: procedure GA_CreateLogFile(FileName : String); external 'dmath';
7673: { Initializes log file }
7674: procedure GenAlg(Func          : TFuncNVar;
7675:                  X, Xmin, Xmax : TVector;
7676:                  Lb, Ub        : Integer;
7677:                  var F_min     : Float); external 'dmath';
7678: { Minimization of a function of several var. by genetic algorithm }
7679: { ---------------------------------------------------------------
7680:   Statistics
7681:   --------------------------------------------------------------- }
7682: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
7683: { Mean of sample X }
7684: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
7685: { Minimum of sample X }
7686: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
7687: { Maximum of sample X }
7688: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
7689: { Median of sample X }
```

```
7690: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
7691: { Standard deviation estimated from sample X }
7692: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
7693: { Standard deviation of population }
7694: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
7695: { Correlation coefficient }
7696: function Skewness(X : TVector; Lb, Ub : Integer; M, Sigma : Float) : Float; external 'dmath';
7697: { Skewness of sample X }
7698: function Kurtosis(X : TVector; Lb, Ub : Integer; M, Sigma : Float) : Float; external 'dmath';
7699: { Kurtosis of sample X }
7700: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
7701: { Quick sort (ascending order) }
7702: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
7703: { Quick sort (descending order) }
7704: procedure Interval(X1, X2             : Float;
7705:                    MinDiv, MaxDiv     : Integer;
7706:                    var Min, Max, Step : Float); external 'dmath';
7707: { Determines an interval for a set of values }
7708: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
7709:                     var XMin, XMax, XStep : Float); external 'dmath';
7710: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
7711: procedure StudIndep(N1, N2         : Integer;
7712:                     M1, M2, S1, S2 : Float;
7713:                     var T          : Float;
7714:                     var DoF        : Integer); external 'dmath';
7715: { Student t-test for independent samples }
7716: procedure StudPaired(X, Y    : TVector;
7717:                      Lb, Ub  : Integer;
7718:                      var T   : Float;
7719:                      var DoF : Integer); external 'dmath';
7720: { Student t-test for paired samples }
7721: procedure AnOVa1(Ns               : Integer;
7722:                  N                : TIntVector;
7723:                  M, S             : TVector;
7724:                  var V_f, V_r, F  : Float;
7725:                  var DoF_f, DoF_r : Integer); external 'dmath';
7726: { One-way analysis of variance }
7727: procedure AnOVa2(NA, NB, Nobs : Integer;
7728:                  M, S         : TMatrix;
7729:                  V, F         : TVector;
7730:                  DoF          : TIntVector); external 'dmath';
7731: { Two-way analysis of variance }
7732: procedure Snedecor(N1, N2         : Integer;
7733:                    S1, S2         : Float;
7734:                    var F          : Float;
7735:                    var DoF1, DoF2 : Integer); external 'dmath';
7736: { Snedecor's F-test (comparison of two variances) }
7737: procedure Bartlett(Ns       : Integer;
7738:                    N        : TIntVector;
7739:                    S        : TVector;
7740:                    var Khi2 : Float;
7741:                    var DoF  : Integer); external 'dmath';
7742: { Bartlett's test (comparison of several variances) }
7743: procedure Mann_Whitney(N1, N2     : Integer;
7744:                        X1, X2     : TVector;
7745:                        var U, Eps : Float); external 'dmath';
7746: { Mann-Whitney test}
7747: procedure Wilcoxon(X, Y       : TVector;
7748:                    Lb, Ub     : Integer;
7749:                    var Ndiff  : Integer;
7750:                    var T, Eps : Float); external 'dmath';
7751: { Wilcoxon test }
7752: procedure Kruskal_Wallis(Ns      : Integer;
7753:                          N       : TIntVector;
7754:                          X       : TMatrix;
7755:                          var H   : Float;
7756:                          var DoF : Integer); external 'dmath';
7757: { Kruskal-Wallis test }
7758: procedure Khi2_Conform(N_cls    : Integer;
7759:                        N_estim  : Integer;
7760:                        Obs      : TIntVector;
7761:                        Calc     : TVector;
7762:                        var Khi2 : Float;
7763:                        var DoF  : Integer); external 'dmath';
7764: { Khi-2 test for conformity }
7765: procedure Khi2_Indep(N_lin    : Integer;
7766:                      N_col    : Integer;
7767:                      Obs      : TIntMatrix;
7768:                      var Khi2 : Float;
7769:                      var DoF  : Integer); external 'dmath';
7770: { Khi-2 test for independence }
7771: procedure Woolf_Conform(N_cls   : Integer;
7772:                         N_estim : Integer;
7773:                         Obs     : TIntVector;
7774:                         Calc    : TVector;
7775:                         var G   : Float;
```

```
7776:                        var DoF : Integer); external 'dmath';
7777: { Woolf's test for conformity }
7778: procedure Woolf_Indep(N_lin   : Integer;
7779:                       N_col   : Integer;
7780:                       Obs     : TIntMatrix;
7781:                       var G   : Float;
7782:                       var DoF : Integer); external 'dmath';
7783: { Woolf's test for independence }
7784: procedure DimStatClassVector(var C : TStatClassVector;
7785:                              Ub    : Integer); external 'dmath';
7786: { Allocates an array of statistical classes: C[0..Ub] }
7787: procedure Distrib(X       : TVector;
7788:                   Lb, Ub  : Integer;
7789:                   A, B, H : Float;
7790:                   C       : TStatClassVector); external 'dmath';
7791: { Distributes an array X[Lb..Ub] into statistical classes }
7792: { ---------------------------------------------------------------
7793:   Linear / polynomial regression
7794:   --------------------------------------------------------------- }
7795: procedure LinFit(X, Y   : TVector;
7796:                  Lb, Ub : Integer;
7797:                  B      : TVector;
7798:                  V      : TMatrix); external 'dmath';
7799: { Linear regression : Y = B(0) + B(1) * X }
7800: procedure WLinFit(X, Y, S : TVector;
7801:                   Lb, Ub  : Integer;
7802:                   B       : TVector;
7803:                   V       : TMatrix); external 'dmath';
7804: { Weighted linear regression : Y = B(0) + B(1) * X }
7805: procedure SVDLinFit(X, Y   : TVector;
7806:                     Lb, Ub : Integer;
7807:                     SVDTol : Float;
7808:                     B      : TVector;
7809:                     V      : TMatrix); external 'dmath';
7810: { Unweighted linear regression by singular value decomposition }
7811: procedure WSVDLinFit(X, Y, S : TVector;
7812:                      Lb, Ub  : Integer;
7813:                      SVDTol  : Float;
7814:                      B       : TVector;
7815:                      V       : TMatrix); external 'dmath';
7816: { Weighted linear regression by singular value decomposition }
7817: procedure MulFit(X            : TMatrix;
7818:                  Y            : TVector;
7819:                  Lb, Ub, Nvar : Integer;
7820:                  ConsTerm     : Boolean;
7821:                  B            : TVector;
7822:                  V            : TMatrix); external 'dmath';
7823: { Multiple linear regression by Gauss-Jordan method }
7824: procedure WMulFit(X            : TMatrix;
7825:                   Y, S         : TVector;
7826:                   Lb, Ub, Nvar : Integer;
7827:                   ConsTerm     : Boolean;
7828:                   B            : TVector;
7829:                   V            : TMatrix); external 'dmath';
7830: { Weighted multiple linear regression by Gauss-Jordan method }
7831: procedure SVDFit(X            : TMatrix;
7832:                  Y            : TVector;
7833:                  Lb, Ub, Nvar : Integer;
7834:                  ConsTerm     : Boolean;
7835:                  SVDTol       : Float;
7836:                  B            : TVector;
7837:                  V            : TMatrix); external 'dmath';
7838: { Multiple linear regression by singular value decomposition }
7839: procedure WSVDFit(X            : TMatrix;
7840:                   Y, S         : TVector;
7841:                   Lb, Ub, Nvar : Integer;
7842:                   ConsTerm     : Boolean;
7843:                   SVDTol       : Float;
7844:                   B            : TVector;
7845:                   V            : TMatrix); external 'dmath';
7846: { Weighted multiple linear regression by singular value decomposition }
7847: procedure PolFit(X, Y        : TVector;
7848:                  Lb, Ub, Deg : Integer;
7849:                  B           : TVector;
7850:                  V           : TMatrix); external 'dmath';
7851: { Polynomial regression by Gauss-Jordan method }
7852: procedure WPolFit(X, Y, S     : TVector;
7853:                   Lb, Ub, Deg : Integer;
7854:                   B           : TVector;
7855:                   V           : TMatrix); external 'dmath';
7856: { Weighted polynomial regression by Gauss-Jordan method }
7857: procedure SVDPolFit(X, Y        : TVector;
7858:                     Lb, Ub, Deg : Integer;
7859:                     SVDTol      : Float;
7860:                     B           : TVector;
7861:                     V           : TMatrix); external 'dmath';
```

```
7862: { Unweighted polynomial regression by singular value decomposition }
7863: procedure WSVDPolFit(X, Y, S     : TVector;
7864:                      Lb, Ub, Deg : Integer;
7865:                      SVDTol      : Float;
7866:                      B           : TVector;
7867:                      V           : TMatrix); external 'dmath';
7868: { Weighted polynomial regression by singular value decomposition }
7869: procedure RegTest(Y, Ycalc : TVector;
7870:                   LbY, UbY : Integer;
7871:                   V        : TMatrix;
7872:                   LbV, UbV : Integer;
7873:                   var Test : TRegTest); external 'dmath';
7874: { Test of unweighted regression }
7875: procedure WRegTest(Y, Ycalc, S : TVector;
7876:                    LbY, UbY    : Integer;
7877:                    V           : TMatrix;
7878:                    LbV, UbV    : Integer;
7879:                    var Test    : TRegTest); external 'dmath';
7880: { Test of weighted regression }
7881: { ---------------------------------------------------------------
7882:   Nonlinear regression
7883:   --------------------------------------------------------------- }
7884: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
7885: { Sets the optimization algorithm for nonlinear regression }
7886: function GetOptAlgo : TOptAlgo; external 'dmath';
7887: { Returns the optimization algorithm }
7888: procedure SetMaxParam(N : Byte); external 'dmath';
7889: { Sets the maximum number of regression parameters for nonlinear regression }
7890: function GetMaxParam : Byte; external 'dmath';
7891: { Returns the maximum number of regression parameters for nonlinear regression }
7892: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
7893: { Sets the bounds on the I-th regression parameter }
7894: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax : Float); external 'dmath';
7895: { Returns the bounds on the I-th regression parameter }
7896: procedure NLFit(RegFunc   : TRegFunc;
7897:                 DerivProc : TDerivProc;
7898:                 X, Y      : TVector;
7899:                 Lb, Ub    : Integer;
7900:                 MaxIter   : Integer;
7901:                 Tol       : Float;
7902:                 B         : TVector;
7903:                 FirstPar,
7904:                 LastPar   : Integer;
7905:                 V         : TMatrix); external 'dmath';
7906: { Unweighted nonlinear regression }
7907: procedure WNLFit(RegFunc   : TRegFunc;
7908:                  DerivProc : TDerivProc;
7909:                  X, Y, S   : TVector;
7910:                  Lb, Ub    : Integer;
7911:                  MaxIter   : Integer;
7912:                  Tol       : Float;
7913:                  B         : TVector;
7914:                  FirstPar,
7915:                  LastPar   : Integer;
7916:                  V         : TMatrix); external 'dmath';
7917: { Weighted nonlinear regression }
7918: procedure SetMCFile(FileName : String); external 'dmath';
7919: { Set file for saving MCMC simulations }
7920: procedure SimFit(RegFunc   : TRegFunc;
7921:                  X, Y      : TVector;
7922:                  Lb, Ub    : Integer;
7923:                  B         : TVector;
7924:                  FirstPar,
7925:                  LastPar   : Integer;
7926:                  V         : TMatrix); external 'dmath';
7927: { Simulation of unweighted nonlinear regression by MCMC }
7928: procedure WSimFit(RegFunc   : TRegFunc;
7929:                  X, Y, S   : TVector;
7930:                  Lb, Ub    : Integer;
7931:                  B         : TVector;
7932:                  FirstPar,
7933:                  LastPar   : Integer;
7934:                  V         : TMatrix); external 'dmath';
7935: { Simulation of weighted nonlinear regression by MCMC }
7936: { ---------------------------------------------------------------
7937:   Nonlinear regression models
7938:   --------------------------------------------------------------- }
7939: procedure FracFit(X, Y       : TVector;
7940:                   Lb, Ub     : Integer;
7941:                   Deg1, Deg2 : Integer;
7942:                   ConsTerm   : Boolean;
7943:                   MaxIter    : Integer;
7944:                   Tol        : Float;
7945:                   B          : TVector;
7946:                   V          : TMatrix); external 'dmath';
7947: { Unweighted fit of rational fraction }
```

```
7948: procedure WFracFit(X, Y, S    : TVector;
7949:                    Lb, Ub     : Integer;
7950:                    Deg1, Deg2 : Integer;
7951:                    ConsTerm   : Boolean;
7952:                    MaxIter    : Integer;
7953:                    Tol        : Float;
7954:                    B          : TVector;
7955:                    V          : TMatrix); external 'dmath';
7956: { Weighted fit of rational fraction }
7957:
7958: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
7959: { Returns the value of the rational fraction at point X }
7960: procedure ExpFit(X, Y         : TVector;
7961:                  Lb, Ub, Nexp : Integer;
7962:                  ConsTerm     : Boolean;
7963:                  MaxIter      : Integer;
7964:                  Tol          : Float;
7965:                  B            : TVector;
7966:                  V            : TMatrix); external 'dmath';
7967: { Unweighted fit of sum of exponentials }
7968: procedure WExpFit(X, Y, S      : TVector;
7969:                   Lb, Ub, Nexp : Integer;
7970:                   ConsTerm     : Boolean;
7971:                   MaxIter      : Integer;
7972:                   Tol          : Float;
7973:                   B            : TVector;
7974:                   V            : TMatrix); external 'dmath';
7975: { Weighted fit of sum of exponentials }
7976: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
7977: { Returns the value of the regression function at point X }
7978: procedure IncExpFit(X, Y     : TVector;
7979:                     Lb, Ub   : Integer;
7980:                     ConsTerm : Boolean;
7981:                     MaxIter  : Integer;
7982:                     Tol      : Float;
7983:                     B        : TVector;
7984:                     V        : TMatrix); external 'dmath';
7985: { Unweighted fit of model of increasing exponential }
7986: procedure WIncExpFit(X, Y, S  : TVector;
7987:                      Lb, Ub   : Integer;
7988:                      ConsTerm : Boolean;
7989:                      MaxIter  : Integer;
7990:                      Tol      : Float;
7991:                      B        : TVector;
7992:                      V        : TMatrix); external 'dmath';
7993: { Weighted fit of increasing exponential }
7994: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
7995: { Returns the value of the regression function at point X }
7996: procedure ExpLinFit(X, Y    : TVector;
7997:                     Lb, Ub  : Integer;
7998:                     MaxIter : Integer;
7999:                     Tol     : Float;
8000:                     B       : TVector;
8001:                     V       : TMatrix); external 'dmath';
8002: { Unweighted fit of the "exponential + linear" model }
8003: procedure WExpLinFit(X, Y, S : TVector;
8004:                      Lb, Ub  : Integer;
8005:                      MaxIter : Integer;
8006:                      Tol     : Float;
8007:                      B       : TVector;
8008:                      V       : TMatrix); external 'dmath';
8009: { Weighted fit of the "exponential + linear" model }
8010:
8011: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
8012: { Returns the value of the regression function at point X }
8013: procedure MichFit(X, Y    : TVector;
8014:                   Lb, Ub  : Integer;
8015:                   MaxIter : Integer;
8016:                   Tol     : Float;
8017:                   B       : TVector;
8018:                   V       : TMatrix); external 'dmath';
8019: { Unweighted fit of Michaelis equation }
8020: procedure WMichFit(X, Y, S : TVector;
8021:                    Lb, Ub  : Integer;
8022:                    MaxIter : Integer;
8023:                    Tol     : Float;
8024:                    B       : TVector;
8025:                    V       : TMatrix); external 'dmath';
8026: { Weighted fit of Michaelis equation }
8027: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
8028: { Returns the value of the Michaelis equation at point X }
8029: procedure MintFit(X, Y    : TVector;
8030:                   Lb, Ub  : Integer;
8031:                   MintVar : TMintVar;
8032:                   Fit_S0  : Boolean;
8033:                   MaxIter : Integer;
```

```
8034:                   Tol     : Float;
8035:                   B       : TVector;
8036:                   V       : TMatrix); external 'dmath';
8037: { Unweighted fit of the integrated Michaelis equation }
8038: procedure WMintFit(X, Y, S : TVector;
8039:                    Lb, Ub  : Integer;
8040:                    MintVar : TMintVar;
8041:                    Fit_S0  : Boolean;
8042:                    MaxIter : Integer;
8043:                    Tol     : Float;
8044:                    B       : TVector;
8045:                    V       : TMatrix); external 'dmath';
8046: { Weighted fit of the integrated Michaelis equation }
8047: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
8048: { Returns the value of the integrated Michaelis equation at point X }
8049: procedure HillFit(X, Y    : TVector;
8050:                   Lb, Ub  : Integer;
8051:                   MaxIter : Integer;
8052:                   Tol     : Float;
8053:                   B       : TVector;
8054:                   V       : TMatrix); external 'dmath';
8055: { Unweighted fit of Hill equation }
8056: procedure WHillFit(X, Y, S : TVector;
8057:                    Lb, Ub  : Integer;
8058:                    MaxIter : Integer;
8059:                    Tol     : Float;
8060:                    B       : TVector;
8061:                    V       : TMatrix); external 'dmath';
8062: { Weighted fit of Hill equation }
8063: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
8064: { Returns the value of the Hill equation at point X }
8065: procedure LogiFit(X, Y     : TVector;
8066:                   Lb, Ub   : Integer;
8067:                   ConsTerm : Boolean;
8068:                   General  : Boolean;
8069:                   MaxIter  : Integer;
8070:                   Tol      : Float;
8071:                   B        : TVector;
8072:                   V        : TMatrix); external 'dmath';
8073: { Unweighted fit of logistic function }
8074: procedure WLogiFit(X, Y, S  : TVector;
8075:                    Lb, Ub   : Integer;
8076:                    ConsTerm : Boolean;
8077:                    General  : Boolean;
8078:                    MaxIter  : Integer;
8079:                    Tol      : Float;
8080:                    B        : TVector;
8081:                    V        : TMatrix); external 'dmath';
8082: { Weighted fit of logistic function }
8083: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
8084: { Returns the value of the logistic function at point X }
8085: procedure PKFit(X, Y    : TVector;
8086:                 Lb, Ub  : Integer;
8087:                 MaxIter : Integer;
8088:                 Tol     : Float;
8089:                 B       : TVector;
8090:                 V       : TMatrix); external 'dmath';
8091: { Unweighted fit of the acid-base titration curve }
8092: procedure WPKFit(X, Y, S : TVector;
8093:                  Lb, Ub  : Integer;
8094:                  MaxIter : Integer;
8095:                  Tol     : Float;
8096:                  B       : TVector;
8097:                  V       : TMatrix); external 'dmath';
8098: { Weighted fit of the acid-base titration curve }
8099: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
8100: { Returns the value of the acid-base titration function at point X }
8101: procedure PowFit(X, Y    : TVector;
8102:                  Lb, Ub  : Integer;
8103:                  MaxIter : Integer;
8104:                  Tol     : Float;
8105:                  B       : TVector;
8106:                  V       : TMatrix); external 'dmath';
8107: { Unweighted fit of power function }
8108: procedure WPowFit(X, Y, S : TVector;
8109:                   Lb, Ub  : Integer;
8110:                   MaxIter : Integer;
8111:                   Tol     : Float;
8112:                   B       : TVector;
8113:                   V       : TMatrix); external 'dmath';
8114: { Weighted fit of power function }
8115:
8116: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
8117: { Returns the value of the power function at point X }
8118: procedure GammaFit(X, Y    : TVector;
8119:                    Lb, Ub  : Integer;
```

```
8120:                   MaxIter : Integer;
8121:                   Tol     : Float;
8122:                   B       : TVector;
8123:                   V       : TMatrix); external 'dmath';
8124: { Unweighted fit of gamma distribution function }
8125: procedure WGammaFit(X, Y, S : TVector;
8126:                    Lb, Ub  : Integer;
8127:                    MaxIter : Integer;
8128:                    Tol     : Float;
8129:                    B       : TVector;
8130:                    V       : TMatrix); external 'dmath';
8131: { Weighted fit of gamma distribution function }
8132: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
8133: { Returns the value of the gamma distribution function at point X }
8134: { ------------------------------------------------------------------
8135:   Principal component analysis
8136:   ------------------------------------------------------------------ }
8137: procedure VecMean(X            : TMatrix;
8138:                   Lb, Ub, Nvar : Integer;
8139:                   M            : TVector); external 'dmath';
8140: { Computes the mean vector M from matrix X }
8141: procedure VecSD(X            : TMatrix;
8142:                 Lb, Ub, Nvar : Integer;
8143:                 M, S         : TVector); external 'dmath';
8144: { Computes the vector of standard deviations S from matrix X }
8145: procedure MatVarCov(X            : TMatrix;
8146:                     Lb, Ub, Nvar : Integer;
8147:                     M            : TVector;
8148:                     V            : TMatrix); external 'dmath';
8149: { Computes the variance-covariance matrix V from matrix X }
8150: procedure MatCorrel(V    : TMatrix;
8151:                     Nvar : Integer;
8152:                     R    : TMatrix); external 'dmath';
8153: { Computes the correlation matrix R from the var-cov matrix V }
8154: procedure PCA(R      : TMatrix;
8155:               Nvar   : Integer;
8156:               Lambda : TVector;
8157:               C, Rc  : TMatrix); external 'dmath';
8158: { Performs a principal component analysis of the correlation matrix R }
8159: procedure ScaleVar(X            : TMatrix;
8160:                    Lb, Ub, Nvar : Integer;
8161:                    M, S         : TVector;
8162:                    Z            : TMatrix); external 'dmath';
8163: { Scales a set of variables by subtracting means and dividing by SD's }
8164: procedure PrinFac(Z            : TMatrix;
8165:                   Lb, Ub, Nvar : Integer;
8166:                   C, F         : TMatrix); external 'dmath';
8167: { Computes principal factors }
8168: { ------------------------------------------------------------------
8169:   Strings
8170:   ------------------------------------------------------------------ }
8171: function LTrim(S : String) : String; external 'dmath';
8172: { Removes leading blanks }
8173: function RTrim(S : String) : String; external 'dmath';
8174: { Removes trailing blanks }
8175: function Trim(S : String) : String; external 'dmath';
8176: { Removes leading and trailing blanks }
8177: function StrChar(N : Byte; C : Char) : String; external 'dmath';
8178: { Returns a string made of character C repeated N times }
8179: function RFill(S : String; L : Byte) : String; external 'dmath';
8180: { Completes string S with trailing blanks for a total length L }
8181: function LFill(S : String; L : Byte) : String; external 'dmath';
8182: { Completes string S with leading blanks for a total length L }
8183: function CFill(S : String; L : Byte) : String; external 'dmath';
8184: { Centers string S on a total length L }
8185: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
8186: { Replaces in string S all the occurences of C1 by C2 }
8187: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
8188: { Extracts a field from a string }
8189: procedure Parse(S : String; Delim : Char; Field : TStrVector; var N : Byte); external 'dmath';
8190: { Parses a string into its constitutive fields }
8191: procedure SetFormat(NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean); external 'dmath';
8192: { Sets the numeric format }
8193: function FloatStr(X : Float) : String; external 'dmath';
8194: { Converts a real to a string according to the numeric format }
8195: function IntStr(N : LongInt) : String; external 'dmath';
8196: { Converts an integer to a string }
8197: function CompStr(Z : Complex) : String; external 'dmath';
8198: { Converts a complex number to a string }
8199: {$IFDEF DELPHI}
8200: function StrDec(S : String) : String; external 'dmath';
8201: { Set decimal separator to the symbol defined in SysUtils }
8202: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
8203: { Test if a string represents a number and returns it in X }
8204: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
8205: { Reads a floating point number from an Edit control }
```

```
8206: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
8207: { Writes a floating point number in a text file }
8208: {$ENDIF}
8209: { ---------------------------------------------------------------
8210:   BGI / Delphi graphics
8211:   --------------------------------------------------------------- }
8212: function InitGraphics
8213: {$IFDEF DELPHI}
8214: (Width, Height : Integer) : Boolean;
8215: {$ELSE}
8216: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
8217: { Enters graphic mode }
8218: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
8219:                     X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
8220: { Sets the graphic window }
8221: procedure SetOxScale(Scale               : TScale;
8222:                      OxMin, OxMax, OxStep : Float); external 'dmath';
8223: { Sets the scale on the Ox axis }
8224: procedure SetOyScale(Scale               : TScale;
8225:                      OyMin, OyMax, OyStep : Float); external 'dmath';
8226: { Sets the scale on the Oy axis }
8227: procedure GetOxScale(var Scale               : TScale;
8228:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
8229: { Returns the scale on the Ox axis }
8230: procedure GetOyScale(var Scale               : TScale;
8231:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
8232: { Returns the scale on the Oy axis }
8233: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
8234: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
8235: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
8236: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
8237: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
8238: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
8239: {$IFNDEF DELPHI}
8240: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
8241: { Sets the font for the main graph title }
8242: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
8243: { Sets the font for the Ox axis (title and labels) }
8244: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
8245: { Sets the font for the Oy axis (title and labels) }
8246: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
8247: { Sets the font for the legends }
8248: procedure SetClipping(Clip : Boolean); external 'dmath';
8249: { Determines whether drawings are clipped at the current viewport
8250:   boundaries, according to the value of the Boolean parameter Clip }
8251: {$ENDIF}
8252: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
8253: { Plots the horizontal axis }
8254: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
8255: { Plots the vertical axis }
8256: procedure PlotGrid({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF} Grid : TGrid); external 'dmath';
8257: { Plots a grid on the graph }
8258: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
8259: { Writes the title of the graph }
8260: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
8261: { Sets the maximum number of curves and re-initializes their parameters }
8262: procedure SetPointParam
8263: {$IFDEF DELPHI}
8264: (CurvIndex, Symbol, Size : Integer; Color : TColor);
8265: {$ELSE}
8266: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
8267: { Sets the point parameters for curve # CurvIndex }
8268: procedure SetLineParam
8269: {$IFDEF DELPHI}
8270: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
8271: {$ELSE}
8272: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
8273: { Sets the line parameters for curve # CurvIndex }
8274: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
8275: { Sets the legend for curve # CurvIndex }
8276: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
8277: { Sets the step for curve # CurvIndex }
8278: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
8279: procedure GetPointParam
8280: {$IFDEF DELPHI}
8281: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
8282: {$ELSE}
8283: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
8284: { Returns the point parameters for curve # CurvIndex }
8285: procedure GetLineParam
8286: {$IFDEF DELPHI}
8287: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
8288: {$ELSE}
8289: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
8290: { Returns the line parameters for curve # CurvIndex }
8291: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
```

```
8292: { Returns the legend for curve # CurvIndex }
8293: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
8294: { Returns the step for curve # CurvIndex }
8295: {$IFDEF DELPHI}
8296: procedure PlotPoint(Canvas    : TCanvas;
8297:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
8298: {$ELSE}
8299: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
8300: {$ENDIF}
8301: { Plots a point on the screen }
8302: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
8303:                     X, Y                  : TVector;
8304:                     Lb, Ub, CurvIndex     : Integer); external 'dmath';
8305: { Plots a curve }
8306: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
8307:                                  X, Y, S               : TVector;
8308:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
8309: { Plots a curve with error bars }
8310: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
8311:                    Func                  : TFunc;
8312:                    Xmin, Xmax            : Float;
8313:                    {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
8314:                    CurvIndex             : Integer); external 'dmath';
8315: { Plots a function }
8316: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
8317:                       NCurv                : Integer;
8318:                       ShowPoints, ShowLines : Boolean); external 'dmath';
8319: { Writes the legends for the plotted curves }
8320: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
8321:                  Nx, Ny, Nc            : Integer;
8322:                  X, Y, Z               : TVector;
8323:                  F                     : TMatrix); external 'dmath';
8324: { Contour plot }
8325: function Xpixel(X : Float) : Integer; external 'dmath'; { Converts user abscissa X to screen coordinate }
8326: function Ypixel(Y : Float) : Integer; external 'dmath'; { Converts user ordinate Y to screen coordinate }
8327: function Xuser(X : Integer) : Float; external 'dmath';  { Converts screen coordinate X to user abscissa }
8328: function Yuser(Y : Integer) : Float; external 'dmath';  { Converts screen coordinate Y to user ordinate }
8329: {$IFNDEF DELPHI}
8330: procedure LeaveGraphics; external 'dmath';
8331: { Quits graphic mode }
8332: {$ENDIF}
8333: { ---------------------------------------------------------------
8334:   LaTeX graphics
8335:   --------------------------------------------------------------- }
8336: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
8337:                           Header            : Boolean) : Boolean; external 'dmath';
8338: { Initializes the LaTeX file }
8339: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
8340: { Sets the graphic window }
8341: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
8342: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
8343: { Sets the scale on the Ox axis }
8344: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
8345: { Sets the scale on the Oy axis }
8346: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
8347: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
8348: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
8349: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
8350: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
8351: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
8352: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
8353: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
8354: { Sets the maximum number of curves and re-initializes their parameters }
8355: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
8356: { Sets the point parameters for curve # CurvIndex }
8357: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
8358:                            Width : Float; Smooth : Boolean); external 'dmath';
8359: { Sets the line parameters for curve # CurvIndex }
8360: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
8361: { Sets the legend for curve # CurvIndex }
8362: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
8363: { Sets the step for curve # CurvIndex }
8364: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
8365: { Plots a curve }
8366: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
8367:                                      Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
8368: { Plots a curve with error bars }
8369: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
8370:                        Npt : Integer; CurvIndex : Integer); external 'dmath';
8371: { Plots a function }
8372: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
8373: { Writes the legends for the plotted curves }
8374: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z    : TVector;
8375:                      F          : TMatrix); external 'dmath';
8376: { Contour plot }
8377: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
```

```
8378: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
8379:
8380: //*****************************************************unit uPSI_SynPdf;
8381:  Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString');
8382:  Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate');
8383:  Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime');
8384:  Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;');
8385:  Function PdfRect1( const Box : TPdfBox) : TPdfRect;');
8386:  Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox');
8387:  //Function _GetCharCount( Text : PAnsiChar) : integer');
8388:  //Procedure L2R( W : PWideChar; L : integer)');
8389:  Function PdfCoord( MM : single) : integer');
8390:  Function CurrentPrinterPaperSize : TPDFPaperSize');
8391:  Function CurrentPrinterRes : TPoint');
8392:  Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)');
8393:  Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)');
8394:  Procedure GDICommentLink( MetaHandle : HDC; const aBookmarkName : RawUTF8; const aRect : TRect)');
8395:  AddConstantN('Usp10','String').SetString( 'usp10.dll');
8396:   AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, f'
8397:    +'InhibitSymSwap, fCharShape, fDigitSubstitute, fInhibitLigate, fDisplayZWG,'
8398:    +' fArabicNumContext, fGcpClusters )');
8399:   AddTypeS('TScriptState_set', 'set of TScriptState_enum');
8400: //*********************************************************************
8401:
8402: procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
8403: begin
8404:  Procedure PMrandomize( I : word)');
8405:  Function PMrandom : longint');
8406:  Function Rrand : extended');
8407:  Function Irand( N : word) : word');
8408:  Function Brand( P : extended) : boolean');
8409:  Function Nrand : extended');
8410: end;
8411:
8412: //*****************************************************unit uPSI_TlHelp32;
8413: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
8414: begin
8415:  AddConstantN('MAX_MODULE_NAME32','LongInt').SetInt( 255);
8416:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle');
8417:  Const('TH32CS_SNAPHEAPLIST','LongWord').SetUInt( $00000001);
8418:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
8419:  AddConstantN('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
8420:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
8421:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
8422:  AddTypeS('tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
8423:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32');
8424:   AddTypeS('THeapList32', 'tagHEAPLIST32');
8425:  Const('HF32_DEFAULT','LongInt').SetInt( 1);
8426:  Const('HF32_SHARED','LongInt').SetInt( 2);
8427:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL');
8428:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL');
8429:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
8430:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
8431:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end');
8432:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32');
8433:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32');
8434:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
8435:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
8436:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
8437:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL');
8438:  Function Heap32Next( var lphe : THeapEntry32) : BOOL');
8439:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL');
8440:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
8441:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
8442:    +'aPri : Longint; dwFlags : DWORD; end');
8443:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32');
8444:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32');
8445:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL');
8446:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL');
8447: end;
8448:
8449:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
8450:  Const('EW_REBOOTSYSTEM','LongWord').SetUInt( $0043);
8451:  Const('EW_EXITANDEXECAPP','LongWord').SetUInt( $0044);
8452:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
8453:  Const('EWX_LOGOFF','LongInt').SetInt( 0);
8454:  Const('EWX_SHUTDOWN','LongInt').SetInt( 1);
8455:  Const('EWX_REBOOT','LongInt').SetInt( 2);
8456:  Const('EWX_FORCE','LongInt').SetInt( 4);
8457:  Const('EWX_POWEROFF','LongInt').SetInt( 8);
8458:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
8459:
8460:   CL.AddTypeS('in_addr', 'record s_bytes : array[1..4] of byte; end;');
8461:  Function socketerror : cint');
8462:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint');
8463:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t');
```

```
8464:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t');
8465:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint');
8466:  Function fplisten( s : cint; backlog : cint) : cint');
8467:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint');
8468:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint');
8469:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint');
8470:  Function NetAddrToStr( Entry : in_addr) : String');
8471:  Function HostAddrToStr( Entry : in_addr) : String');
8472:  Function StrToHostAddr( IP : String) : in_addr');
8473:  Function StrToNetAddr( IP : String) : in_addr');
8474:  CL.AddConstantN('SOL_SOCKET','LongWord').SetUInt( $ffff);
8475:  CL.AddTypeS('cint8', 'shortint');
8476:   CL.AddTypeS('cuint8', 'byte');
8477:   CL.AddTypeS('cchar', 'cint8');
8478:   CL.AddTypeS('cschar', 'cint8');
8479:   CL.AddTypeS('cuchar', 'cuint8');
8480:   CL.AddTypeS('cint16', 'smallint');
8481:   CL.AddTypeS('cuint16', 'word');
8482:   CL.AddTypeS('cshort', 'cint16');
8483:   CL.AddTypeS('csshort', 'cint16');
8484:   CL.AddTypeS('cushort', 'cuint16');
8485:   CL.AddTypeS('cint32', 'longint');
8486:   CL.AddTypeS('cuint32', 'longword');
8487:   CL.AddTypeS('cint', 'cint32');
8488:   CL.AddTypeS('csint', 'cint32');
8489:   CL.AddTypeS('cuint', 'cuint32');
8490:   CL.AddTypeS('csigned', 'cint');
8491:   CL.AddTypeS('cunsigned', 'cuint');
8492:   CL.AddTypeS('cint64', 'int64');
8493:   CL.AddTypeS('clonglong', 'cint64');
8494:   CL.AddTypeS('cslonglong', 'cint64');
8495:   CL.AddTypeS('cbool', 'longbool');
8496:   CL.AddTypeS('cfloat', 'single');
8497:   CL.AddTypeS('cdouble', 'double');
8498:   CL.AddTypeS('clongdouble', 'extended');
8499:
8500:
8501: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
8502: begin
8503:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint');
8504:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean');
8505:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean');
8506:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean');
8507:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean');
8508:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer');
8509:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer');
8510:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string');
8511:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string');
8512:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64');
8513:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64');
8514:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)');
8515:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)');
8516:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)');
8517:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)');
8518:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean');
8519:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean');
8520:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean');
8521:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean');
8522:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
8523:    +'RunOnce, ekServiceRun, ekServiceRunOnce )');
8524:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError');
8525:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean');
8526:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean');
8527:  Function RegSaveList(const RootKey:HKEY; const Key: string; const ListName: string;const Items:TStrings):
      Boolean');
8528:  Function RegLoadList(const RootKey:HKEY; const Key: string; const ListName: string;const SaveTo:
      TStrings):Boolean');
8529:  Function RegDelList( const RootKey : HKEY; const Key : string; const ListName : string) : Boolean');
8530: end;
8531:
8532: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
8533: begin
8534:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
8535:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
8536:  KelvinFreezingPoint','Extended').setExtended( 273.15);
8537:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
8538:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
8539:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
8540:  DegPerCycle','Extended').setExtended( 360.0);
8541:  DegPerGrad','Extended').setExtended( 0.9);
8542:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
8543:  GradPerCycle','Extended').setExtended( 400.0);
8544:  GradPerDeg','Extended').setExtended( 1.111111111111111111111111111111);
8545:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
8546:  RadPerCycle','Extended').setExtended( 6.283185307179586476925286766559);
8547:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
```

```
8548:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
8549:  CyclePerDeg','Extended').setExtended( 0.0027777777777777777777777777777778);
8550:  CyclePerGrad','Extended').setExtended( 0.0025);
8551:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
8552:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
8553:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
8554:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint');
8555:  Function MakePercentage( const Step, Max : Longint) : Longint');
8556:  Function CelsiusToKelvin( const T : double) : double');
8557:  Function CelsiusToFahrenheit( const T : double) : double');
8558:  Function KelvinToCelsius( const T : double) : double');
8559:  Function KelvinToFahrenheit( const T : double) : double');
8560:  Function FahrenheitToCelsius( const T : double) : double');
8561:  Function FahrenheitToKelvin( const T : double) : double');
8562:  Function CycleToDeg( const Cycles : double) : double');
8563:  Function CycleToGrad( const Cycles : double) : double');
8564:  Function CycleToRad( const Cycles : double) : double');
8565:  Function DegToCycle( const Degrees : double) : double');
8566:  Function DegToGrad( const Degrees : double) : double');
8567:  Function DegToRad( const Degrees : double) : double');
8568:  Function GradToCycle( const Grads : double) : double');
8569:  Function GradToDeg( const Grads : double) : double');
8570:  Function GradToRad( const Grads : double) : double');
8571:  Function RadToCycle( const Radians : double) : double');
8572:  Function RadToDeg( const Radians : double) : double');
8573:  Function RadToGrad( const Radians : double) : double');
8574:  Function DmsToDeg( const D, M : Integer; const S : double) : double');
8575:  Function DmsToRad( const D, M : Integer; const S : double) : double');
8576:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)');
8577:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string');
8578:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)');
8579:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)');
8580:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)');
8581:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)');
8582:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)');
8583:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)');
8584:  Function CmToInch( const Cm : double) : double');
8585:  Function InchToCm( const Inch : double) : double');
8586:  Function FeetToMetre( const Feet : double) : double');
8587:  Function MetreToFeet( const Metre : double) : double');
8588:  Function YardToMetre( const Yard : double) : double');
8589:  Function MetreToYard( const Metre : double) : double');
8590:  Function NmToKm( const Nm : double) : double');
8591:  Function KmToNm( const Km : double) : double');
8592:  Function KmToSm( const Km : double) : double');
8593:  Function SmToKm( const Sm : double) : double');
8594:  Function LitreToGalUs( const Litre : double) : double');
8595:  Function GalUsToLitre( const GalUs : double) : double');
8596:  Function GalUsToGalCan( const GalUs : double) : double');
8597:  Function GalCanToGalUs( const GalCan : double) : double');
8598:  Function GalUsToGalUk( const GalUs : double) : double');
8599:  Function GalUkToGalUs( const GalUk : double) : double');
8600:  Function LitreToGalCan( const Litre : double) : double');
8601:  Function GalCanToLitre( const GalCan : double) : double');
8602:  Function LitreToGalUk( const Litre : double) : double');
8603:  Function GalUkToLitre( const GalUk : double) : double');
8604:  Function KgToLb( const Kg : double) : double');
8605:  Function LbToKg( const Lb : double) : double');
8606:  Function KgToOz( const Kg : double) : double');
8607:  Function OzToKg( const Oz : double) : double');
8608:  Function CwtUsToKg( const Cwt : double) : double');
8609:  Function CwtUkToKg( const Cwt : double) : double');
8610:  Function KaratToKg( const Karat : double) : double');
8611:  Function KgToCwtUs( const Kg : double) : double');
8612:  Function KgToCwtUk( const Kg : double) : double');
8613:  Function KgToKarat( const Kg : double) : double');
8614:  Function KgToSton( const Kg : double) : double');
8615:  Function KgToLton( const Kg : double) : double');
8616:  Function StonToKg( const STon : double) : double');
8617:  Function LtonToKg( const Lton : double) : double');
8618:  Function QrUsToKg( const Qr : double) : double');
8619:  Function QrUkToKg( const Qr : double) : double');
8620:  Function KgToQrUs( const Kg : double) : double');
8621:  Function KgToQrUk( const Kg : double) : double');
8622:  Function PascalToBar( const Pa : double) : double');
8623:  Function PascalToAt( const Pa : double) : double');
8624:  Function PascalToTorr( const Pa : double) : double');
8625:  Function BarToPascal( const Bar : double) : double');
8626:  Function AtToPascal( const At : double) : double');
8627:  Function TorrToPascal( const Torr : double) : double');
8628:  Function KnotToMs( const Knot : double) : double');
8629:  Function HpElectricToWatt( const HpE : double) : double');
8630:  Function HpMetricToWatt( const HpM : double) : double');
8631:  Function MsToKnot( const ms : double) : double');
8632:  Function WattToHpElectric( const W : double) : double');
8633:  Function WattToHpMetric( const W : double) : double');
```

```
8634:  function getBigPI: string;');    //PI of 1000 numbers
8635:
8636: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
8637: begin
8638:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle');
8639:  Procedure CDCopyFile( const FileName, DestName : string)');
8640:  Procedure CDMoveFile( const FileName, DestName : string)');
8641:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor');
8642:  Procedure CDDeleteFiles( Sender : TObject; s : string)');
8643:  Function CDGetTempDir : string');
8644:  Function CDGetFileSize( FileName : string) : longint');
8645:  Function GetFileTime( FileName : string) : longint');
8646:  Function GetShortName( FileName : string) : string');
8647:  Function GetFullName( FileName : string) : string');
8648:  Function WinReboot : boolean');
8649:  Function WinDir : String');
8650:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal');
8651:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean');
8652:  Function devExecutor : TdevExecutor');
8653:
8654: end;
8655:
8656: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
8657: begin
8658:  Procedure CheckAssociations');
8659:  Procedure Associate( Index : integer)');
8660:  Procedure UnAssociate( Index : integer)');
8661:  Function IsAssociated( Index : integer) : boolean');
8662:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean');
8663:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp, IcoNum : string)');
8664:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)');
8665:  CL.AddConstantN('AssociationsCount','LongInt').SetInt( 7);
8666:  procedure RefreshIcons;
8667:
8668: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
8669: function MergColor(Colors: Array of TColor): TColor;
8670: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
8671: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
8672: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
8673: function GetInverseColor(AColor: TColor): TColor;
8674: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
8675: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X, Y: integer;
8676:   ShadowColor: TColor);
8677: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
8678: Procedure GetSystemMenuFont(Font: TFont);
8679: end;
8680:
8681: //**************************unit uPSI_JvHLParser;*******************************
8682: function IsStringConstant(const St: string): Boolean;
8683: function IsIntConstant(const St: string): Boolean;
8684: function IsRealConstant(const St: string): Boolean;
8685: function IsIdentifier(const ID: string): Boolean;
8686: function GetStringValue(const St: string): string;
8687: procedure ParseString(const S: string; Ss: TStrings);
8688:
8689: function IsStringConstantW(const St: WideString): Boolean;
8690: function IsIntConstantW(const St: WideString): Boolean;
8691: function IsRealConstantW(const St: WideString): Boolean;
8692: function IsIdentifierW(const ID: WideString): Boolean;
8693: function GetStringValueW(const St: WideString): WideString;
8694: procedure ParseStringW(const S: WideString; Ss: TStrings);
8695:
8696:
8697: //**************************unit uPSI_JclMapi;*******************************
8698:
8699: Function JclSimpleSendMail( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
      TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean');
8700: Function JclSimpleSendFax( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
      TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean');
8701: Function JclSimpleBringUpSendMailDialog( const ASubject, ABody : string; const AAttachment : TFileName;
      AParentWND : HWND) : Boolean');
8702: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD');
8703: Function MapiErrorMessage( const ErrorCode : DWORD) : string');
8704:
8705:
8706:
8707: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
8708: begin
8709:  AddConstantN('ExtPascalVersion','String').SetString( '0.9.8');
8710:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
8711:    +'Opera, brKonqueror, brMobileSafari )');
8712:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )');
8713:   AddTypeS('TExtProcedure', 'Procedure');
8714:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser');
8715:  Function ExtExtract(const Delims:array of string;var S:string;var
      Matches:TStringList;Remove:boolean):boolean');
```

```
8716:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList');
8717:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer');
8718:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer');
8719:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer');
8720:  Function StrToJS( const S : string; UseBR : boolean) : string');
8721:  Function CaseOf( const S : string; const Cases : array of string) : integer');
8722:  Function RCaseOf( const S : string; const Cases : array of string) : integer');
8723:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string');
8724:  Function SetPaddings( Top : integer; Right : integer; Bottom : integer; Left : integer; CSSUnit :
      TCSSUnit; Header : boolean) : string');
8725:  Function SetMargins( Top : integer; Right : integer; Bottom : integer; Left : integer; CSSUnit :
      TCSSUnit; Header : boolean) : string');
8726:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean');
8727:  Function IsUpperCase( S : string) : boolean');
8728:  Function BeautifyJS( const AScript : string; const StartingLevel : integer; SplitHTMLNewLine : boolean) :
      string');
8729:  Function BeautifyCSS( const AStyle : string) : string');
8730:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer');
8731:  Function JSDateToDateTime( JSDate : string) : TDateTime');
8732: end;
8733:
8734: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
8735: begin
8736:   CL.AddTypeS('TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )');
8737:   CL.AddTypeS('TSHDeleteOptions', 'set of TSHDeleteOption');
8738:   CL.AddTypeS('TSHRenameOption', '( roSilent, roRenameOnCollision )');
8739:   CL.AddTypeS('TSHRenameOptions', 'set of TSHRenameOption');
8740:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean');
8741:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean');
8742:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean');
8743:   CL.AddTypeS('TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )');
8744:   CL.AddTypeS('TEnumFolderFlags', 'set of TEnumFolderFlag');
8745:   CL.AddTypeS('TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
8746:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
8747:    +'IEnumIdList; Folder : IShellFolder; end');
8748:  Function SHEnumFolderFirst( const Folder : string; Flags : TEnumFolderFlags; var F : TEnumFolderRec) :
      Boolean');
8749:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD; Flags:TEnumFolderFlags;var F : TEnumFolderRec) :
      Boolean');
8750:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)');
8751:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean');
8752:  Function GetSpecialFolderLocation( const Folder : Integer) : string');
8753:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;');
8754:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;');
8755:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean');
8756:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean');
8757:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean');
8758:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean');
8759:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean');
8760:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean');
8761:  Function SHFreeMem( var P : Pointer) : Boolean');
8762:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList');
8763:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList');
8764:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList');
8765:  Function PidlBindToParent( const IdList : PItemIdList; out Folder : IShellFolder; out Last : PItemIdList)
      : Boolean');
8766:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean');
8767:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean');
8768:  Function PidlFree( var IdList : PItemIdList) : Boolean');
8769:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer');
8770:  Function PidlGetLength( const Pidl : PItemIdList) : Integer');
8771:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList');
8772:  Function PidlToPath( IdList : PItemIdList) : string');
8773:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean');
8774:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string');
8775:   CL.AddTypeS('PShellLink', '^TShellLink // will not work');
8776:   CL.AddTypeS('TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
8777:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
8778:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end');
8779:  Procedure ShellLinkFree( var Link : TShellLink)');
8780:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT');
8781:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT');
8782:  Function ShellLinkCreateSystem( const Link : TShellLink; const Folder : Integer; const FileName : string)
      : HRESULT');
8783:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean');
8784:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean');
8785:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON');
8786:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean');
8787:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean');
8788:  Function OverlayIconShared( var Large, Small : HICON) : Boolean');
8789:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string');
8790:  Function ShellExecEx(const FileName:string; const Parameters:string;const Verb: string; CmdShow:
      Integer): Boolean');
8791:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directory:
      string;ShowCommand:Integer):Boolean');
8792:  Function ShellExecAndWait(const FileName:string;const Parameters: string;const
      Verb:string;CmdShow:Integer): Boolean);
```

```
8793:  Function ShellOpenAs( const FileName : string) : Boolean');
8794:  Function ShellRasDial( const EntryName : string) : Boolean');
8795:  Function ShellRunControlPanel( const NameOrFileName : string; AppletNumber : Integer) : Boolean');
8796:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON');
8797:   CL.AddTypeS('TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )');
8798:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType');
8799:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string');
8800: end;
8801:
8802:
8803:
8804: Functions_max hex in the box maXbox
8805: functionslist.txt
8806: FunctionsList1 3.9.9
8807:
8808: ********************************************************************************
8809: Procedure
8810: PROCEDURE SIZE 3797 3600 3385 3296 2883 2255 (2065) (1854)
8811: Procedure ***************************Now the Procedure list*******************
8812: Procedure ( ACol, ARow : Integer; Items : TStrings)
8813: Procedure ( Agg : TAggregate)
8814: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
8815: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
8816: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
8817: Procedure ( ASender : TObject; const ABytes : Integer)
8818: Procedure ( ASender : TObject; VStream : TStream)
8819: Procedure ( AThread : TIdThread)
8820: Procedure ( AWebModule : TComponent)
8821: Procedure ( Column : TColumn)
8822: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
8823: Procedure ( const iStart : integer; const sText : string)
8824: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
8825: Procedure ( Database : TDatabase; LoginParams : TStrings)
8826: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:
      TReconcileAction)
8827: Procedure ( DATASET : TDATASET)
8828: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
8829: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
8830: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
8831: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
8832: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
8833: Procedure ( Done : Integer)
8834: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
8835: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
8836: Procedure
      (HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
8837: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
8838: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
8839: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
8840: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
8841: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
8842: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
8843: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
8844: Procedure ( SENDER : TFIELD; const TEXT : String)
8845: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
8846: Procedure ( Sender : TIdTelnet; const Buffer : String)
8847: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
8848: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
8849: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
8850: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
8851: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
8852: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
8853: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
8854: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
8855: Procedure ( Sender : TObject; Button : TMPBtnType)
8856: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
8857: Procedure ( Sender : TObject; Button : TUDBtnType)
8858: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
8859: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
8860: Procedure ( Sender : TObject; Column : TListColumn)
8861: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
8862: Procedure ( Sender : TObject; Connecting : Boolean)
8863: Procedure ( Sender:TObject; const PaperSize:SmallInt;const Orientation:TPrinterOrientation;const
      PageType:TPageType;var DoneDrawing: Boolean)
8864: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
8865: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
8866: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
8867: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
8868: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
8869: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
8870: Procedure ( Sender : TObject; Index : LongInt)
8871: Procedure ( Sender : TObject; Item : TListItem)
8872: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
8873: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
8874: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
8875: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
```

```
8876: Procedure ( Sender : TObject; Item : TListItem; var S : string)
8877: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
8878: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
8879: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
8880: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
8881: Procedure ( Sender : TObject; Node : TTreeNode)
8882: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
8883: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
8884: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
8885: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
8886: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
8887: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
8888: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
8889: Procedure ( Sender : TObject; Rect : TRect)
8890: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
8891: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
8892: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
8893: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
8894: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
8895: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
8896: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
8897: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
8898: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
8899: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
8900: Procedure ( Sender : TObject; Thread : TServerClientThread)
8901: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
8902: Procedure ( Sender : TObject; Username, Password : string)
8903: Procedure ( Sender : TObject; var AllowChange : Boolean)
8904: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction : TUpDownDirection)
8905: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
8906: Procedure ( Sender : TObject; var Continue : Boolean)
8907: Procedure ( Sender : TObject; var dest: string; var NumRedirect: Integer; var Handled: boolean; var
      VMethod : TIdHTTPMethod)
8908: Procedure ( Sender : TObject; var Username : string)
8909: Procedure ( Sender : TObject; Wnd : HWND)
8910: Procedure ( Sender : TToolbar; Button : TToolButton)
8911: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
8912: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
8913: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
8914: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
8915: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
8916: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
8917: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
8918: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
8919: procedure (Sender: TObject)
8920: procedure (Sender: TObject; var Done: Boolean)
8921: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
8922: procedure _T(Name: tbtString; v: Variant);
8923: Procedure AbandonSignalHandler( RtlSigNum : Integer)
8924: Procedure Abort
8925: Procedure About1Click( Sender : TObject)
8926: Procedure Accept( Socket : TSocket)
8927: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
8928: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
8929: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
8930: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
8931: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
8932: Procedure Add( Addend1, Addend2 : TMyBigInt)
8933: Procedure ADD( const AKEY, AVALUE : VARIANT)
8934: Procedure Add( const Key : string; Value : Integer)
8935: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
8936: Procedure ADD( FIELD : TFIELD)
8937: Procedure ADD( ITEM : TMENUITEM)
8938: Procedure ADD( POPUP : TPOPUPMENU)
8939: Procedure AddCharacters( xCharacters : TCharSet)
8940: Procedure AddDriver( const Name : string; List : TStrings)
8941: Procedure AddImages( Value : TCustomImageList)
8942: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
8943: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
8944: Procedure AddLoader( Loader : TBitmapLoader)
8945: Procedure ADDPARAM( VALUE : TPARAM)
8946: Procedure AddPassword( const Password : string)
8947: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
8948: Procedure AddState( oState : TniRegularExpressionState)
8949: Procedure AddStrings( Strings : TStrings);
8950: procedure AddStrings(Strings: TStrings);
8951: Procedure AddStrings1( Strings : TWideStrings);
8952: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
8953: Procedure AddToRecentDocs( const Filename : string)
8954: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
8955: Procedure AllFunctionsList1Click( Sender : TObject)
8956: procedure AllObjectsList1Click(Sender: TObject);
8957: Procedure Allocate( AAllocateBytes : Integer)
8958: procedure AllResourceList1Click(Sender: TObject);
8959: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
8960: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
```

```
8961: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
8962: Procedure AnsiFree( var s : AnsiString)
8963: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
8964: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
8965: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
8966: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
8967: Procedure AntiFreeze;
8968: Procedure APPEND
8969: Procedure Append( const S : WideString)
8970: procedure Append(S: string);
8971: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
8972: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
8973: Procedure AppendChunk( Val : OleVariant)
8974: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
8975: Procedure AppendStr( var Dest : string; S : string)
8976: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
8977: Procedure ApplyRange
8978: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
8979: Procedure Arrange( Code : TListArrangement)
8980: procedure Assert(expr : Boolean; const msg: string);
8981: procedure Assert2(expr : Boolean; const msg: string);
8982: Procedure Assign( AList : TCustomBucketList)
8983: Procedure Assign( Other : TObject)
8984: Procedure Assign( Source : TDragObject)
8985: Procedure Assign( Source : TPersistent)
8986: Procedure Assign(Source: TPersistent)
8987: procedure Assign2(mystring, mypath: string);
8988: Procedure AssignCurValues( Source : TDataSet);
8989: Procedure AssignCurValues1( const CurValues : Variant);
8990: Procedure ASSIGNFIELD( FIELD : TFIELD)
8991: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
8992: Procedure AssignFile(var F: Text; FileName: string)
8993: procedure AssignFile(var F: TextFile; FileName: string)
8994: procedure AssignFileRead(var mystring, myfilename: string);
8995: procedure AssignFileWrite(mystring, myfilename: string);
8996: Procedure AssignTo( Other : TObject)
8997: Procedure AssignValues( Value : TParameters)
8998: Procedure ASSIGNVALUES( VALUE : TPARAMS)
8999: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
9000: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
9001: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
9002: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
9003: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
9004: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
9005: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
9006: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
9007: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
9008: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
9009: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
9010: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
9011: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
9012: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
9013: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
9014: procedure Beep
9015: Procedure BeepOk');
9016: Procedure BeepQuestion');
9017: Procedure BeepHand');
9018: Procedure BeepExclamation');
9019: Procedure BeepAsterisk');
9020: Procedure BeepInformation');
9021: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
9022: Procedure BeginLayout
9023: Procedure BeginTimer( const Delay, Resolution : Cardinal)
9024: Procedure BeginUpdate
9025: procedure BeginUpdate;
9026: procedure BigScreen1Click(Sender: TObject);
9027: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
9028: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
9029: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
9030: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
9031: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
9032: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
9033: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
9034: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
9035: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
9036: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
9037: Procedure BreakPointMenuClick( Sender : TObject)
9038: procedure BRINGTOFRONT
9039: procedure BringToFront;
9040: Procedure btnBackClick( Sender : TObject)
9041: Procedure btnBrowseClick( Sender : TObject)
9042: Procedure BtnClick( Index : TNavigateBtn)
9043: Procedure btnLargeIconsClick( Sender : TObject)
9044: Procedure BuildAndSendRequest( AURI : TIdURI)
9045: Procedure BuildCache
9046: Procedure BurnMemory( var Buff, BuffLen : integer)
```

```
9047: Procedure BurnMemoryStream( Destructo : TMemoryStream)
9048: Procedure CalculateFirstSet
9049: Procedure Cancel
9050: procedure CancelDrag;
9051: Procedure CancelEdit
9052: procedure CANCELHINT
9053: Procedure CancelRange
9054: Procedure CancelUpdates
9055: Procedure CancelWriteBuffer
9056: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Boolean;
9057: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
9058: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Boolean
9059: procedure CaptureScreenFormat(vname: string; vextension: string);
9060: procedure CaptureScreenPNG(vname: string);
9061: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
9062: procedure CASCADE
9063: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
9064: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
9065: Procedure cbPathClick( Sender : TObject)
9066: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
9067: Procedure cedebugAfterExecute( Sender : TPSScript)
9068: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
9069: Procedure cedebugCompile( Sender : TPSScript)
9070: Procedure cedebugExecute( Sender : TPSScript)
9071: Procedure cedebugIdle( Sender : TObject)
9072: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
9073: Procedure CenterHeight( const pc, pcParent : TControl)
9074: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
9075: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
9076: Procedure Change
9077: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
9078: Procedure Changed
9079: Procedure ChangeDir( const ADirName : string)
9080: Procedure ChangeDirUp
9081: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
9082: Procedure ChangeLevelBy( Value : TChangeRange)
9083: Procedure ChDir(const s: string)
9084: Procedure Check(Status: Integer)
9085: Procedure CheckCommonControl( CC : Integer)
9086: Procedure CHECKFIELDNAME( const FIELDNAME : String)
9087: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
9088: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:boolean)
9089: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
9090: Procedure CheckToken( T : Char)
9091: procedure CheckToken(t:char)
9092: Procedure CheckTokenSymbol( const S : string)
9093: procedure CheckTokenSymbol(s:string)
9094: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
9095: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
9096: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
9097: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
9098: procedure CipherFile1Click(Sender: TObject);
9099: Procedure Clear;
9100: Procedure Clear1Click( Sender : TObject)
9101: Procedure ClearColor( Color : TColor)
9102: Procedure CLEARITEM( AITEM : TMENUITEM)
9103: Procedure ClearMapping
9104: Procedure ClearSelection( KeepPrimary : Boolean)
9105: Procedure ClearWriteBuffer
9106: Procedure Click
9107: Procedure Close
9108: Procedure Close1Click( Sender : TObject)
9109: Procedure CloseDatabase( Database : TDatabase)
9110: Procedure CloseDataSets
9111: Procedure CloseDialog
9112: Procedure CloseFile(var F: Text);
9113: Procedure Closure
9114: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
9115: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
9116: Procedure CodeCompletionList1Click( Sender : TObject)
9117: Procedure ColEnter
9118: Procedure Collapse
9119: Procedure Collapse( Recurse : Boolean)
9120: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
9121: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
9122: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
9123: Procedure Compile1Click( Sender : TObject)
9124: procedure ComponentCount1Click(Sender: TObject);
9125: Procedure Compress(azipfolder, azipfile: string)');
9126: Procedure DeCompress(azipfolder, azipfile: string)');
9127: Procedure XZip(azipfolder, azipfile: string)');
9128: Procedure XUnZip(azipfolder, azipfile: string)');
9129: Procedure Connect( const ATimeout : Integer)
9130: Procedure Connect( Socket : TSocket)
9131: procedure Console1Click(Sender: TObject);
9132: Procedure Continue
```

```
9133: Procedure ContinueCount( var Counter : TJclCounter)
9134: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
9135: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
9136: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
9137: Procedure ConvertImage(vsource, vdestination: string);
9138: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
9139: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
9140: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
9141: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
9142: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
9143: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
9144: Procedure CopyFrom( mbCopy : TMyBigInt)
9145: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
9146: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
9147: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
      of Byte; const ADestIndex : Integer; const ALength : Integer)
9148: Procedure CopyTIdBytes( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : TIdBytes;
      const ADestIndex : Integer; const ALength : Integer)
9149: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
9150: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
9151: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
9152: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
9153: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
9154: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
9155: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
9156: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
9157: Procedure CopyToClipboard
9158: Procedure CountParts
9159: Procedure CreateDataSet
9160: Procedure CreateEmptyFile( const FileName : string)
9161: Procedure CreateFileFromString( const FileName, Data : string)
9162: Procedure CreateFromDelta( Source : TPacketDataSet)
9163: procedure CREATEHANDLE
9164: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
9165: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
9166: Procedure CreateTable
9167: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
9168: procedure CSyntax1Click(Sender: TObject);
9169: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
9170: Procedure CURSORPOSCHANGED
9171: procedure CutFirstDirectory(var S: String)
9172: Procedure DataBaseError(const Message: string)
9173: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
9174: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
9175: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
9176: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
9177: Procedure DBIError(errorCode: Integer)
9178: Procedure DebugOutput( const AText : string)
9179: Procedure DebugRun1Click( Sender : TObject)
9180: procedure Dec;
9181: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
9182: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
9183: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
9184: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
9185: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
9186: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
9187: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
9188: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
9189: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
9190: Procedure Decompile1Click( Sender : TObject)
9191: Procedure DefaultDrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State :
      TGridDrawState)
9192: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
9193: Procedure DeferLayout
9194: Procedure defFileread
9195: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
9196: Procedure Delete
9197: Procedure Delete( const AFilename : string)
9198: Procedure Delete( const Index : Integer)
9199: Procedure DELETE( INDEX : INTEGER)
9200: Procedure Delete( Index : LongInt)
9201: Procedure Delete( Node : TTreeNode)
9202: procedure Delete(var s: AnyString; ifrom, icount: Longint);
9203: Procedure DeleteAlias( const Name : string)
9204: Procedure DeleteDriver( const Name : string)
9205: Procedure DeleteIndex( const Name : string)
9206: Procedure DeleteKey( const Section, Ident : String)
9207: Procedure DeleteRecords
9208: Procedure DeleteRecords( AffectRecords : TAffectRecords)
9209: Procedure DeleteString( var pStr : String; const pDelStr : string)
9210: Procedure DeleteTable
9211: procedure DelphiSite1Click(Sender: TObject);
9212: Procedure Deselect
9213: Procedure Deselect( Node : TTreeNode)
9214: procedure DestroyComponents
9215: Procedure DestroyHandle
```

```
9216: Procedure Diff( var X : array of Double)
9217: procedure Diff(var X: array of Double);
9218: procedure DISABLEALIGN
9219: Procedure DisableConstraints
9220: Procedure Disconnect
9221: Procedure Disconnect( Socket : TSocket)
9222: Procedure Dispose
9223: procedure Dispose(P: PChar)
9224: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
9225: Procedure DoKey( Key : TDBCtrlGridKey)
9226: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
9227: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
9228: Procedure Dormant
9229: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
9230: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
9231: Procedure DoubleToComp( Value : Double; var Result : Comp)
9232: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
9233: procedure Draw(X, Y: Integer; Graphic: TGraphic);
9234: Procedure Draw1(Canvas:TCanvas; X,Y,
      Index:Integer;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Boolean);
9235: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
9236: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
9237: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
9238: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
9239: procedure DrawFocusRect(const Rect: TRect);
9240: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
9241: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
9242: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
9243: Procedure DrawOverlay1( Canvas : TCanvas; X, Y : Integer; ImageIndex : Integer; Overlay : TOverlay;
      ADrawingStyle : TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
9244: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
9245: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
9246: Procedure DropConnections
9247: Procedure DropDown
9248: Procedure DumpDescription( oStrings : TStrings)
9249: Procedure DumpStateTable( oStrings : TStrings)
9250: Procedure EDIT
9251: Procedure EditButtonClick
9252: Procedure EditFont1Click( Sender : TObject)
9253: procedure Ellipse(X1, Y1, X2, Y2: Integer);
9254: Procedure Ellipse1( const Rect : TRect);
9255: Procedure EMMS
9256: Procedure Encode( ADest : TStream)
9257: procedure ENDDRAG(DROP:BOOLEAN)
9258: Procedure EndEdit( Cancel : Boolean)
9259: Procedure EndTimer
9260: Procedure EndUpdate
9261: Procedure EraseSection( const Section : string)
9262: Procedure Error( const Ident : string)
9263: procedure Error(Ident:Integer)
9264: Procedure ErrorFmt( const Ident : string; const Args : array of const)
9265: Procedure ErrorStr( const Message : string)
9266: procedure ErrorStr(Message:String)
9267: Procedure Exchange( Index1, Index2 : Integer)
9268: procedure Exchange(Index1, Index2: Integer);
9269: Procedure Exec( FileName, Parameters, Directory : string)
9270: Procedure ExecProc
9271: Procedure ExecSQL( UpdateKind : TUpdateKind)
9272: Procedure Execute
9273: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
9274: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
9275: Procedure ExecuteCommand(executeFile, paramstring: string)
9276: Procedure ExecuteShell(executeFile, paramstring: string)
9277: Procedure ExitThread(ExitCode: Integer); stdcall;');
9278: Procedure ExitProcess(ExitCode: Integer); stdcall;');
9279: Procedure Expand( AUserName : String; AResults : TStrings)
9280: Procedure Expand( Recurse : Boolean)
9281: Procedure ExportClipboard1Click( Sender : TObject)
9282: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)');
9283: Procedure ExtractContentFields( Strings : TStrings)
9284: Procedure ExtractCookieFields( Strings : TStrings)
9285: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
9286: Procedure ExtractHeaderFields(Separators,WhiteSpace: TSysCharSet; Content: PChar; Strings: TStrings;
      Decode: Boolean; StripQuotes : Boolean)
9287: Procedure ExtractHTTPFields(Separators,WhiteSpace: TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes
      : Boolean)
9288: Procedure ExtractQueryFields( Strings : TStrings)
9289: Procedure FastDegToGrad
9290: Procedure FastDegToRad
9291: Procedure FastGradToDeg
9292: Procedure FastGradToRad
9293: Procedure FastRadToDeg
9294: Procedure FastRadToGrad
9295: Procedure FileClose( Handle : Integer)
9296: Procedure FileClose(handle: integer)
9297: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Boolean)
```

```
9298: Procedure FileStructure( AStructure : TIdFTPDataStructure)
9299: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
9300: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
9301: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
9302: Procedure FillChar2(var X: PChar ; count: integer; value: char)
9303: Procedure FillCharS(var p: string; count: integer; value: char);'); //fix3.8
9304: Procedure FillIPList
9305: procedure FillRect(const Rect: TRect);
9306: Procedure FillTStrings( AStrings : TStrings)
9307: Procedure FilterOnBookmarks( Bookmarks : array of const)
9308: procedure FinalizePackage(Module: HMODULE)
9309: procedure FindClose;
9310: procedure FindClose2(var F: TSearchRec)
9311: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
9312: Procedure FindMatches1( const sString : string; iStart : integer; xNotify :
      TniRegularExpressionMatchFoundEvent);
9313: Procedure FindNearest( const KeyValues : array of const)
9314: Procedure FinishContext
9315: Procedure FIRST
9316: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
9317: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extended;ValueType:TFloatValue;Precision,
      Decimals:Integer);
9318: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
9319: Procedure FlushSchemaCache( const TableName : string)
9320: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
9321: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
9322: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
9323: Procedure FormActivate( Sender : TObject)
9324: procedure FormatLn(const format: String; const args: array of const);');  //alias
9325: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
9326: Procedure FormCreate( Sender : TObject)
9327: Procedure FormDestroy( Sender : TObject)
9328: Procedure FormKeyPress( Sender : TObject; var Key : Char)
9329: procedure FormOutput1Click(Sender: TObject);
9330: Procedure FormToHtml( Form : TForm; Path : string)');
9331: procedure FrameRect(const Rect: TRect);
9332: Procedure Frame3D( Canvas : TCanvas; var Rect : TRect; TopColor, BottomColor : TColor; Width : Integer)');
9333: Procedure NotebookHandlesNeeded( Notebook : TNotebook)');
9334: Procedure Free( Buffer : TRecordBuffer)
9335: Procedure Free( Buffer : TValueBuffer)
9336: Procedure Free;
9337: Procedure FreeAndNil(var Obj:TObject)
9338: Procedure FreeImage
9339: procedure FreeMem(P: PChar; Size: Integer)
9340: Procedure FreeTreeData( Tree : TUpdateTree)
9341: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
9342: Procedure FullCollapse
9343: Procedure FullExpand
9344: Procedure Get1( AURL : string; const AResponseContent : TStream);
9345: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
9346: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
9347: Procedure GetAliasNames( List : TStrings)
9348: Procedure GetAliasParams( const AliasName : string; List : TStrings)
9349: Procedure GetApplicationsRunning( Strings : TStrings)
9350: Procedure GetCommandTypes( List : TWideStrings)
9351: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
9352: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
9353: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
9354: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
9355: Procedure GetDatabaseNames( List : TStrings)
9356: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
9357: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
9358: Procedure GetDir(d: byte; var s: string)
9359: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
9360: Procedure GetDriverNames( List : TStrings)
9361: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
9362: Procedure GetDriverParams( const DriverName : string; List : TStrings)
9363: Procedure GetEMails1Click( Sender : TObject)
9364: Procedure getEnvironmentInfo;
9365: Function getEnvironmentString: string;
9366: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
9367: Procedure GetFieldNames( const TableName : string; List : TStrings)
9368: Procedure GetFieldNames( const TableName : string; List : TStrings);
9369: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
9370: Procedure GETFIELDNAMES( LIST : TSTRINGS)
9371: Procedure GetFieldNames1( const TableName : string; List : TStrings);
9372: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
9373: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings);
9374: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
9375: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
9376: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
9377: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
9378: Procedure GetFormatSettings
9379: Procedure GetFromDIB( var DIB : TBitmapInfo)
9380: Procedure GetFromHDIB( HDIB : HBitmap)
9381: Procedure GetIcon( Index : Integer; Image : TIcon);
```

```
9382: Procedure GetIcon1(Index : Integer; Image : TIcon; ADrawingStyle : TDrawingStyle; AImageType : TImageType);
9383: Procedure GetIndexInfo( IndexName : string)
9384: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
9385: Procedure GetIndexNames( List : TStrings)
9386: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
9387: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
9388: Procedure GetIndexNames4( const TableName : string; List : TStrings);
9389: Procedure GetInternalResponse
9390: Procedure GETITEMNAMES( LIST : TSTRINGS)
9391: procedure GetMem(P: PChar; Size: Integer)
9392: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
9393: procedure GetPackageDescription(ModuleName: PChar): string)
9394: Procedure GetPackageNames( List : TStrings);
9395: Procedure GetPackageNames1( List : TWideStrings);
9396: Procedure GetParamList( List : TList; const ParamNames : WideString)
9397: Procedure GetProcedureNames( List : TStrings);
9398: Procedure GetProcedureNames( List : TWideStrings);
9399: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
9400: Procedure GetProcedureNames1( List : TStrings);
9401: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
9402: Procedure GetProcedureNames3( List : TWideStrings);
9403: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
9404: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
9405: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
9406: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
9407: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
9408: Procedure GetProviderNames( Names : TWideStrings);
9409: Procedure GetProviderNames( Proc : TGetStrProc)
9410: Procedure GetProviderNames1( Names : TStrings);
9411: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
9412: Procedure GetSchemaNames( List : TStrings);
9413: Procedure GetSchemaNames1( List : TWideStrings);
9414: Procedure GetSessionNames( List : TStrings)
9415: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
9416: Procedure GetStrings( List : TStrings)
9417: Procedure GetSystemTime; stdcall;
9418: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
9419: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
9420: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
9421: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
9422: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
9423: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
9424: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
9425: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
9426: Procedure GetVisibleWindows( List : Tstrings)
9427: Procedure GoBegin
9428: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
9429: Procedure GotoCurrent( Table : TTable)
9430: procedure GotoEnd1Click(Sender: TObject);
9431: Procedure GotoNearest
9432: Procedure GradientFillCanvas( const ACanvas : TCanvas; const AStartColor, AEndColor : TColor; const ARect
      : TRect; const Direction : TGradientDirection)
9433: Procedure HandleException( E : Exception; var Handled : Boolean)
9434: procedure HANDLEMESSAGE
9435: procedure HandleNeeded;
9436: Procedure Head( AURL : string)
9437: Procedure Help( var AHelpContents : TStringList; ACommand : String)
9438: Procedure HexToBinary( Stream : TStream)
9439: procedure HexToBinary(Stream:TStream)
9440: Procedure HideDragImage
9441: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
9442: Procedure HideTraybar
9443: Procedure HookOSExceptions
9444: Procedure HookSignal( RtlSigNum : Integer)
9445: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
9446: Procedure HTMLSyntax1Click( Sender : TObject)
9447: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
9448: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
9449: Procedure ImportfromClipboard1Click( Sender : TObject)
9450: Procedure ImportfromClipboard2Click( Sender : TObject)
9451: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
9452: procedure Incb(var x: byte);
9453: Procedure Include1Click( Sender : TObject)
9454: Procedure IncludeOFF;  //preprocessing
9455: Procedure IncludeON;
9456: procedure Info1Click(Sender: TObject);
9457: Procedure InitAltRecBuffers( CheckModified : Boolean)
9458: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
9459: Procedure InitContext(WebModuleList: TAbstractWebModuleList;Request: TWebRequest; Response : TWebResponse)
9460: Procedure InitData( ASource : TDataSet)
9461: Procedure InitDelta( ADelta : TPacketDataSet);
9462: Procedure InitDelta1( const ADelta : OleVariant);
9463: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
9464: Procedure Initialize
9465: procedure InitializePackage(Module: HMODULE)
9466: Procedure INITIATEACTION
```

```
9467: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
9468: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
9469: Procedure InitModule( AModule : TComponent)
9470: Procedure InitStdConvs
9471: Procedure InitTreeData( Tree : TUpdateTree)
9472: Procedure INSERT
9473: Procedure Insert( Index : Integer; AClass : TClass)
9474: Procedure Insert( Index : Integer; AComponent : TComponent)
9475: Procedure Insert( Index : Integer; AObject : TObject)
9476: Procedure Insert( Index : Integer; const S : WideString)
9477: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
9478: Procedure Insert(Index: Integer; const S: string);
9479: procedure Insert(Index: Integer; S: string);
9480: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
9481: procedure InsertComponent(AComponent:TComponent)
9482: procedure InsertControl(AControl: TControl);
9483: Procedure InsertIcon( Index : Integer; Image : TIcon)
9484: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
9485: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
9486: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
9487: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
9488: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
9489: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
9490: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
9491: Procedure InternalBeforeResolve( Tree : TUpdateTree)
9492: Procedure InvalidateModuleCache
9493: Procedure InvalidateTitles
9494: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
9495: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
9496: Procedure InvalidDateTimeError(const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond : Word;
      const ABaseDate : TDateTime)
9497: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
9498: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
9499: procedure JavaSyntax1Click(Sender: TObject);
9500: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)');
9501: Procedure KillDataChannel
9502: Procedure Largefont1Click( Sender : TObject)
9503: Procedure LAST
9504: Procedure LaunchCpl( FileName : string)
9505: Procedure Launch( const AFile : string)');
9506: Procedure LaunchFile( const AFile : string)');
9507: Procedure lineToNumber( xmemo : String; met : boolean)
9508: Procedure ListViewCustomDrawItem(Sender:TCustomListView;Item:TListItem; State:TCustomDrawState;var
      DefaultDraw:Boolean)
9509: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
      : TCustomDrawState; var DefaultDraw : Boolean)
9510: Procedure ListViewData( Sender : TObject; Item : TListItem)
9511: Procedure ListViewDataFind( Sender : TObject; Find : TItemFind; const FindString : String; const
      FindPosition : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var
      Index: Integer)
9512: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
9513: Procedure ListViewDblClick( Sender : TObject)
9514: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
9515: Procedure ListDLLExports(const FileName: string; List: TStrings);
9516: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
9517: procedure LoadBytecode1Click(Sender: TObject);
9518: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
9519: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
9520: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
9521: Procedure LoadFromFile( AFileName : string)
9522: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
9523: Procedure LoadFromFile( const FileName : string)
9524: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
9525: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
9526: Procedure LoadFromFile( const FileName : WideString)
9527: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
9528: Procedure LoadFromFile(const AFileName: string)
9529: procedure LoadFromFile(FileName: string);
9530: procedure LoadFromFile(FileName:String)
9531: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
9532: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
9533: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
9534: Procedure LoadFromStream( const Stream : TStream)
9535: Procedure LoadFromStream( S : TStream)
9536: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
9537: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
9538: Procedure LoadFromStream( Stream : TStream)
9539: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
9540: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
9541: procedure LoadFromStream(Stream: TStream);
9542: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
9543: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
9544: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
9545: Procedure LoadLastFile1Click( Sender : TObject)
9546: { LoadIcoToImage loads two icons from resource named NameRes,
9547:   into two image lists ALarge and ASmall}
```

```
9548: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
9549: Procedure LoadMemo
9550: Procedure LoadParamsFromIniFile( FFileName : WideString)
9551: Procedure Lock
9552: Procedure Login
9553: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
9554: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
9555: Procedure MakeCaseInsensitive
9556: Procedure MakeDeterministic( var bChanged : boolean)
9557: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
9558: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
9559: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
9560: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:boolean;Volume:Byte);
9561: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);');
9562: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);');
9563: Procedure MakeVisible
9564: Procedure MakeVisible( PartialOK : Boolean)
9565: Procedure Manual1Click( Sender : TObject)
9566: Procedure MarkReachable
9567: Procedure maXbox;  //shows the exe version data in a win box
9568: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
9569: Procedure Memo1Change( Sender : TObject)
9570: Procedure Memo1ReplaceText(Sender : TObject; const ASearch, AReplace : String; Line, Column : Integer; var
      Action : TSynReplaceAction)
9571: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
9572: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
9573: procedure Memory1Click(Sender: TObject);
9574: Procedure MERGE( MENU : TMAINMENU)
9575: Procedure MergeChangeLog
9576: procedure MINIMIZE
9577: Procedure MinimizeMaxbox;
9578: Procedure MkDir(const s: string)
9579: Procedure mnuPrintFont1Click( Sender : TObject)
9580: procedure ModalStarted
9581: Procedure Modified
9582: Procedure ModifyAlias( Name : string; List : TStrings)
9583: Procedure ModifyDriver( Name : string; List : TStrings)
9584: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
9585: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
9586: Procedure Move( CurIndex, NewIndex : Integer)
9587: procedure Move(CurIndex, NewIndex: Integer);
9588: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
9589: Procedure MoveChars(const ASource: String; ASourceStart:integer;var ADest: String; ADestStart,ALen:integer)
9590: Procedure moveCube( o : TMyLabel)
9591: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
9592: procedure MoveTo(X, Y: Integer);
9593: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
9594: Procedure MovePoint(var x,y:Extended; const angle:Extended);
9595: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
9596: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
9597: Procedure MsgAbout(Handle: Integer;const Msg,Caption: string; const IcoName:string = 'MAINICON';
      Flags:DWORD = MB_OK);
9598: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);');
9599: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
9600: procedure New(P: PChar)
9601: procedure New1Click(Sender: TObject);
9602: procedure NewInstance1Click(Sender: TObject);
9603: Procedure NEXT
9604: Procedure NextMonth
9605: Procedure Noop
9606: Procedure NormalizePath( var APath : string)
9607: procedure ObjectBinaryToText(Input, Output: TStream)
9608: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
9609: procedure ObjectResourceToText(Input, Output: TStream)
9610: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
9611: procedure ObjectTextToBinary(Input, Output: TStream)
9612: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
9613: procedure ObjectTextToResource(Input, Output: TStream)
9614: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
9615: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
9616: Procedure Open( const UserID : WideString; const Password : WideString);
9617: Procedure Open;
9618: Procedure open1Click( Sender : TObject)
9619: Procedure OpenCdDrive
9620: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
9621: Procedure OpenCurrent
9622: Procedure OpenFile(vfilenamepath: string)');
9623: Procedure OpenDirectory1Click( Sender : TObject)
9624: Procedure OpenIndexFile( const IndexName : string)
9625: Procedure OpenSchema( const Schema : TSchemaInfo; const Restrictions : OleVariant; const SchemaID :
      OleVariant; DataSet : TADODataSet)
9626: Procedure OpenWriteBuffer( const AThreshhold : Integer)
9627: Procedure OptimizeMem
9628: Procedure Options1( AURL : string);
9629: Procedure OutputDebugString(lpOutputString : PChar)');
9630: Procedure PackBuffer
```

```
9631: Procedure Paint
9632: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
9633: Procedure PaintToTBitmap( Target : TBitmap)
9634: Procedure PaletteChanged
9635: Procedure ParentBiDiModeChanged
9636: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
9637: Procedure PasteFromClipboard;
9638: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
9639: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
9640: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
9641: Procedure PError( Text : string)
9642: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
9643: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Integer;Y3:Integer;X4:Integer;Y4:Integer);
9644: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
9645: procedure playmp3(mpath: string);
9646: Procedure PlayMP31Click( Sender : TObject)
9647: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
9648: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
9649: procedure PolyBezier(const Points: array of TPoint);
9650: procedure PolyBezierTo(const Points: array of TPoint);
9651: procedure Polygon(const Points: array of TPoint);
9652: procedure Polyline(const Points: array of TPoint);
9653: Procedure Pop
9654: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INTEGER; var WIDTHS : array of LONGINT)
9655: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
9656: Procedure POPUP( X, Y : INTEGER)
9657: Procedure PopupURL(URL : WideString);
9658: Procedure POST
9659: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
9660: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
9661: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
9662: Procedure PostUser( const Email, FirstName, LastName : WideString)
9663: procedure Pred(X: int64);
9664: Procedure Prepare
9665: Procedure PrepareStatement
9666: Procedure PreProcessXML( AList : TStrings)');
9667: Procedure PreventDestruction
9668: Procedure Print( const Caption : string)
9669: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
9670: procedure printf(const format: String; const args: array of const);
9671: Procedure PrintList(Value: TStringList);');
9672: Procedure PrintImage(aValue: TBitmap; Style:TBitmapStyle);)//TBitmapStyle=(bsNormal,bsCentered,
     bsStretched);
9673: Procedure Printout1Click( Sender : TObject)
9674: Procedure ProcessHeaders
9675: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
9676: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
9677: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
9678: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
9679: Procedure ProcessMessagesOFF;  //application.processmessages
9680: Procedure ProcessMessagesON;
9681: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
9682: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
9683: Procedure Proclist Size is: 3797 /1415
9684: Procedure procMessClick( Sender : TObject)
9685: Procedure PSScriptCompile( Sender : TPSScript)
9686: Procedure PSScriptExecute( Sender : TPSScript)
9687: Procedure PSScriptLine( Sender : TObject)
9688: Procedure Push( ABoundary : string)
9689: procedure PushItem(AItem: Pointer)
9690: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
9691: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
9692: procedure PutLinuxLines(const Value: string)
9693: Procedure Quit
9694: Procedure RaiseConversionError( const AText : string);
9695: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
9696: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
9697: procedure RaiseException(Ex: TIFException; Param: String);
9698: Procedure RaiseExceptionForLastCmdResult;
9699: procedure RaiseLastException;
9700: procedure RaiseException2;
9701: Procedure RaiseLastOSError
9702: Procedure RaiseLastWin32;
9703: procedure RaiseLastWin32Error)
9704: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
9705: Procedure RandomFillStream( Stream : TMemoryStream)
9706: procedure randomize;
9707: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)');
9708: Procedure RCS
9709: Procedure Read( Socket : TSocket)
9710: Procedure ReadBlobData
9711: procedure ReadBuffer(Buffer:String;Count:LongInt)
9712: procedure ReadOnly1Click(Sender: TObject);
9713: Procedure ReadSection( const Section : string; Strings : TStrings)
9714: Procedure ReadSections( Strings : TStrings)
9715: Procedure ReadSections( Strings : TStrings);
```

```
9716: Procedure ReadSections1( const Section : string; Strings : TStrings);
9717: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
9718: Procedure ReadStream( AStream : TStream; AByteCount : LongInt; const AReadUntilDisconnect : boolean)
9719: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
9720: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
9721: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
9722: Procedure Realign;
9723: procedure Rectangle(X1, Y1, X2, Y2: Integer);
9724: Procedure Rectangle1( const Rect : TRect);
9725: Procedure RectCopy( var Dest : TRect; const Source : TRect)
9726: Procedure RectFitToScreen( var R : TRect)
9727: Procedure RectGrow( var R : TRect; const Delta : Integer)
9728: Procedure RectGrowX( var R : TRect; const Delta : Integer)
9729: Procedure RectGrowY( var R : TRect; const Delta : Integer)
9730: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
9731: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
9732: Procedure RectNormalize( var R : TRect)
9733: Procedure RedirectTransition( oOldState : TniRegularExpressionState; oNewState : TniRegularExpressionState)
9734: Procedure Refresh;
9735: Procedure RefreshData( Options : TFetchOptions)
9736: Procedure REFRESHLOOKUPLIST
9737: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
9738: Procedure RegisterChanges( Value : TChangeLink)
9739: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
9740: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
9741: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Integer)
9742: Procedure ReInitialize( ADelay : Cardinal)
9743: procedure RELEASE
9744: Procedure Remove( const AByteCount : integer)
9745: Procedure REMOVE( FIELD : TFIELD)
9746: Procedure REMOVE( ITEM : TMENUITEM)
9747: Procedure REMOVE( POPUP : TPOPUPMENU)
9748: Procedure RemoveAllPasswords
9749: procedure RemoveComponent(AComponent:TComponent)
9750: Procedure RemoveDir( const ADirName : string)
9751: Procedure RemoveLambdaTransitions( var bChanged : boolean)
9752: Procedure REMOVEPARAM( VALUE : TPARAM)
9753: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
9754: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
9755: Procedure Rename( const ASourceFile, ADestFile : string)
9756: Procedure Rename( const FileName : string; Reload : Boolean)
9757: Procedure RenameTable( const NewTableName : string)
9758: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
9759: Procedure Replace1Click( Sender : TObject)
9760: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
9761: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
9762: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
9763: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
9764: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
9765: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
9766: Procedure Requery( Options : TExecuteOptions)
9767: Procedure Reset
9768: Procedure Reset1Click( Sender : TObject)
9769: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
9770: procedure ResourceExplore1Click(Sender: TObject);
9771: Procedure RestoreContents
9772: Procedure RestoreDefaults
9773: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
9774: Procedure RetrieveHeaders
9775: Procedure RevertRecord
9776: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
9777: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
9778: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
9779: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
9780: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
9781: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
9782: Procedure RleCompress2( Stream : TStream)');
9783: Procedure RleDecompress2( Stream : TStream)');
9784: Procedure RmDir(const S: string)
9785: Procedure Rollback
9786: Procedure Rollback( TransDesc : TTransactionDesc)
9787: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
9788: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
9789: Procedure RollbackTrans
9790: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
9791: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
9792: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
9793: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
9794: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
9795: Procedure S_EBox( const AText : string)
9796: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:integer;var MultKey:integer;var
      AddKey:integer
9797: Procedure S_IBox( const AText : string)
9798: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
9799: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
9800: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
```

```
9801: Procedure SampleVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
9802: Procedure Save2Click( Sender : TObject)
9803: Procedure Saveas3Click( Sender : TObject)
9804: Procedure Savebefore1Click( Sender : TObject)
9805: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
9806: Procedure SaveConfigFile
9807: Procedure SaveOutput1Click( Sender : TObject)
9808: procedure SaveScreenshotClick(Sender: TObject);
9809: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
9810: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
9811: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
9812: Procedure SaveToFile( AFileName : string)
9813: Procedure SAVETOFILE( const FILENAME : String)
9814: Procedure SaveToFile( const FileName : WideString)
9815: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
9816: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
9817: procedure SaveToFile(FileName: string);
9818: procedure SaveToFile(FileName:String)
9819: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
9820: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
9821: Procedure SaveToStream( S : TStream)
9822: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
9823: Procedure SaveToStream( Stream : TStream)
9824: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
9825: procedure SaveToStream(Stream: TStream);
9826: procedure SaveToStream(Stream:TStream)
9827: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
9828: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
9829: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
9830: procedure Say(const sText: string)
9831: Procedure SBytecode1Click( Sender : TObject)
9832: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
9833: procedure ScriptExplorer1Click(Sender: TObject);
9834: Procedure Scroll( Distance : Integer)
9835: Procedure Scroll( DX, DY : Integer)
9836: procedure ScrollBy(DeltaX, DeltaY: Integer);
9837: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
9838: Procedure ScrollTabs( Delta : Integer)
9839: Procedure Search1Click( Sender : TObject)
9840: procedure SearchAndOpenDoc(vfilenamepath: string)
9841: procedure SearchAndOpenFile(vfilenamepath: string)
9842: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
9843: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
9844: Procedure SearchNext1Click( Sender : TObject)
9845: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
9846: Procedure Select1( const Nodes : array of TTreeNode);
9847: Procedure Select2( Nodes : TList);
9848: Procedure SelectNext( Direction : Boolean)
9849: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
9850: Procedure SelfTestPEM');  //unit uPSI_cPEM
9851: Procedure Send( AMsg : TIdMessage)
9852: //config forst in const MAILINIFILE = 'maildef.ini';
9853: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','');
9854: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
9855: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
9856: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
9857: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
9858: Procedure SendResponse
9859: Procedure SendStream( AStream : TStream)
9860: Procedure Set8087CW( NewCW : Word)
9861: Procedure SetAll( One, Two, Three, Four : Byte)
9862: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
9863: Procedure SetAppDispatcher( const ADispatcher : TComponent)
9864: procedure SetArrayLength;
9865: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
9866: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
9867: Procedure SetAsHandle( Format : Word; Value : THandle)
9868: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
9869: procedure SetCaptureControl(Control: TControl);
9870: Procedure SetColumnAttributes
9871: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
9872: Procedure SetCustomHeader( const Name, Value : string)
9873: Procedure SetExprParams(const Text: Widestring;Options:TFilterOptions;ParserOptions:TParserOptions; const
      FieldName: Widestring)
9874: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
9875: Procedure SetFocus
9876: procedure SetFocus; virtual;
9877: Procedure SetInitialState
9878: Procedure SetKey
9879: procedure SetLastError(ErrorCode: Integer)
9880: procedure SetLength;
9881: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
9882: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
9883: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
9884: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
9885: Procedure SetParams1( UpdateKind : TUpdateKind);
```

```
9886: Procedure SetPassword( const Password : string)
9887: Procedure SetPointer( Ptr : Pointer; Size : Longint)
9888: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
9889: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
9890: Procedure SetProvider( Provider : TComponent)
9891: Procedure SetProxy( const Proxy : string)
9892: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
9893: Procedure SetRange( const StartValues, EndValues : array of const)
9894: Procedure SetRangeEnd
9895: Procedure SetRate( const aPercent, aYear : integer)
9896: procedure SetRate(const aPercent, aYear: integer)
9897: Procedure SetSafeCallExceptionMsg( Msg : String)
9898: procedure SETSELTEXTBUF(BUFFER:PCHAR)
9899: Procedure SetSize( AWidth, AHeight : Integer)
9900: procedure SetSize(NewSize:LongInt)
9901: procedure SetString(var s: string; buffer: PChar; len: Integer)
9902: Procedure SetStrings( List : TStrings)
9903: Procedure SetText( Text : PwideChar)
9904: procedure SetText(Text: PChar);
9905: Procedure SetTextBuf( Buffer : PChar)
9906: procedure SETTEXTBUF(BUFFER:PCHAR)
9907: Procedure SetTick( Value : Integer)
9908: Procedure SetTimeout( ATimeOut : Integer)
9909: Procedure SetTraceEvent( Event : TDBXTraceEvent)
9910: Procedure SetUserName( const UserName : string)
9911: Procedure SetWallpaper( Path : string);
9912: procedure ShellStyle1Click(Sender: TObject);
9913: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
9914: Procedure ShowFileProperties( const FileName : string)
9915: Procedure ShowInclude1Click( Sender : TObject)
9916: Procedure ShowInterfaces1Click( Sender : TObject)
9917: Procedure ShowLastException1Click( Sender : TObject)
9918: Procedure ShowMessage( const Msg : string)
9919: Procedure ShowMessageBig(const aText : string);
9920: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
9921: Procedure MsgBig(const aText : string);        //alias
9922: procedure showmessage(mytext: string);
9923: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
9924: procedure ShowMessageFmt(const Msg: string; Params: array of const))
9925: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
9926: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
9927: Procedure ShowSearchDialog( const Directory : string)
9928: Procedure ShowSpecChars1Click( Sender : TObject)
9929: Procedure ShredFile( const FileName : string; Times : Integer)
9930: procedure Shuffle(vQ: TStringList);
9931: Procedure ShuffleList( var List : array of Integer; Count : Integer)
9932: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
9933: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
9934: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
9935: Procedure Site( const ACommand : string)
9936: Procedure SkipEOL
9937: Procedure Sleep( ATime : cardinal)
9938: Procedure Sleep( milliseconds : Cardinal)
9939: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD');
9940: Procedure Slinenumbers1Click( Sender : TObject)
9941: Procedure Sort
9942: Procedure SortColorArray(ColorArray:TColorArray;L,R:Integer;SortType:TColorArraySortType;Reverse: Boolean)
9943: procedure Speak(const sText: string)   //async like voice
9944: procedure Speak2(const sText: string)  //sync
9945: procedure Split(Str: string;  SubStr: string; List: TStrings);
9946: Procedure SplitNameValue( const Line : string; var Name, Value : string)');
9947: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
9948: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
9949: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
9950: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
9951: procedure SQLSyntax1Click(Sender: TObject);
9952: Procedure Start
9953: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
9954: Procedure StartTransaction( TransDesc : TTransactionDesc)
9955: Procedure Status( var AStatusList : TStringList)
9956: Procedure StatusBar1DblClick( Sender : TObject)
9957: Procedure StepInto1Click( Sender : TObject)
9958: Procedure StepIt
9959: Procedure StepOut1Click( Sender : TObject)
9960: Procedure Stop
9961: procedure stopmp3;
9962: Procedure StrDispose( Str : PChar)
9963: procedure StrDispose(Str: PChar)
9964: Procedure StrReplace(var Str: String; Old, New: String);
9965: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
9966: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
9967: Procedure StringToBytes( Value : String; Bytes : array of byte)
9968: procedure StrSet(c : Char; I : Integer; var s : String);
9969: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
9970: Procedure StructureMount( APath : String)
9971: procedure STYLECHANGED(SENDER:TOBJECT)
```

```
 9972: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
 9973: procedure Succ(X: int64);
 9974: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
 9975: procedure SwapChar(var X,Y: char);  //swapX follows
 9976: Procedure SwapFloats( var X, Y : Float)
 9977: procedure SwapGrid(grd: TStringGrid);
 9978: Procedure SwapOrd( var I, J : Byte);
 9979: Procedure SwapOrd( var X, Y : Integer)
 9980: Procedure SwapOrd1( var I, J : Shortint);
 9981: Procedure SwapOrd2( var I, J : Smallint);
 9982: Procedure SwapOrd3( var I, J : Word);
 9983: Procedure SwapOrd4( var I, J : Integer);
 9984: Procedure SwapOrd5( var I, J : Cardinal);
 9985: Procedure SwapOrd6( var I, J : Int64);
 9986: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
 9987: Procedure Synchronize1( Method : TMethod);
 9988: procedure SyntaxCheck1Click(Sender: TObject);
 9989: Procedure SysFreeString(const S: WideString); stdcall;
 9990: Procedure TakeOver( Other : TLinearBitmap)
 9991: Procedure Talkln(const sText: string)');  //async voice
 9992: procedure tbtn6resClick(Sender: TObject);
 9993: Procedure tbtnUseCaseClick( Sender : TObject)
 9994: procedure TerminalStyle1Click(Sender: TObject);
 9995: Procedure Terminate
 9996: Procedure texSyntax1Click( Sender : TObject)
 9997: procedure TextOut(X, Y: Integer; Text: string);
 9998: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
 9999: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
10000: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
10001: Procedure TextStart
10002: procedure TILE
10003: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
10004: Procedure TitleClick( Column : TColumn)
10005: Procedure ToDo
10006: procedure toolbtnTutorialClick(Sender: TObject);
10007: Procedure Trace1( AURL : string; const AResponseContent : TStream);
10008: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
10009: Procedure Truncate
10010: procedure Tutorial101Click(Sender: TObject);
10011: procedure Tutorial10Statistics1Click(Sender: TObject);
10012: procedure Tutorial11Forms1Click(Sender: TObject);
10013: procedure Tutorial12SQL1Click(Sender: TObject);
10014: Procedure tutorial1Click( Sender : TObject)
10015: Procedure tutorial21Click( Sender : TObject)
10016: Procedure tutorial31Click( Sender : TObject)
10017: Procedure tutorial4Click( Sender : TObject)
10018: Procedure Tutorial5Click( Sender : TObject)
10019: procedure Tutorial6Click(Sender: TObject);
10020: procedure Tutorial91Click(Sender: TObject);
10021: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
10022: procedure UniqueString(var str: AnsiString)
10023: procedure UnloadLoadPackage(Module: HMODULE)
10024: Procedure Unlock
10025: Procedure UNMERGE( MENU : TMAINMENU)
10026: Procedure UnRegisterChanges( Value : TChangeLink)
10027: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
10028: Procedure UnregisterConversionType( const AType : TConvType)
10029: Procedure UnRegisterProvider( Prov : TCustomProvider)
10030: Procedure UPDATE
10031: Procedure UpdateBatch( AffectRecords : TAffectRecords)
10032: Procedure UPDATECURSORPOS
10033: Procedure UpdateFile
10034: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
10035: Procedure UpdateResponse( AResponse : TWebResponse)
10036: Procedure UpdateScrollBar
10037: Procedure UpdateView1Click( Sender : TObject)
10038: procedure Val(const s: string; var n, z: Integer)
10039: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
10040: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
10041: Procedure VariantAdd( const src : Variant; var dst : Variant)
10042: Procedure VariantAnd( const src : Variant; var dst : Variant)
10043: Procedure VariantArrayRedim( var V : Variant; High : Integer)
10044: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
10045: Procedure VariantClear( var V : Variant)
10046: Procedure VariantCpy( const src : Variant; var dst : Variant)
10047: Procedure VariantDiv( const src : Variant; var dst : Variant)
10048: Procedure VariantMod( const src : Variant; var dst : Variant)
10049: Procedure VariantMul( const src : Variant; var dst : Variant)
10050: Procedure VariantOr( const src : Variant; var dst : Variant)
10051: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
10052: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
10053: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
10054: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
10055: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
10056: Procedure VariantShl( const src : Variant; var dst : Variant)
10057: Procedure VariantShr( const src : Variant; var dst : Variant)
```

```
10058: Procedure VariantSub( const src : Variant; var dst : Variant)
10059: Procedure VariantXor( const src : Variant; var dst : Variant)
10060: Procedure VarCastError;');
10061: Procedure VarCastError1( const ASourceType, ADestType : TVarType);');
10062: Procedure VarInvalidOp');
10063: Procedure VarInvalidNullOp');
10064: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)');
10065: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)');
10066: Procedure VarArrayCreateError');
10067: Procedure VarResultCheck( AResult : HRESULT);');
10068: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);');
10069: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)');
10070: Function VarTypeAsText( const AType : TVarType) : string');
10071: procedure Voice(const sText: string)//async
10072: procedure Voice2(const sText: string)');//sync
10073: Procedure WaitMiliSeconds( AMSec : word)
10074: Procedure WideAppend( var dst : WideString; const src : WideString)
10075: Procedure WideAssign( var dst : WideString; var src : WideString)
10076: Procedure WideDelete( var dst : WideString; index, count : Integer)
10077: Procedure WideFree( var s : WideString)
10078: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
10079: Procedure WideFromPChar( var dst : WideString; src : PChar)
10080: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
10081: Procedure WideSetLength( var dst : WideString; len : Integer)
10082: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
10083: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
10084: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
10085: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
10086: Procedure WordWrap1Click( Sender : TObject)
10087: Procedure Write( const AOut : string)
10088: Procedure Write( Socket : TSocket)
10089: procedure Write(S: string);
10090: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
10091: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
10092: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
10093: procedure WriteBuffer(Buffer:String;Count:LongInt)
10094: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
10095: Procedure WriteChar( AValue : Char)
10096: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
10097: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
10098: Procedure WriteFloat( const Section, Name : string; Value : Double)
10099: Procedure WriteHeader( AHeader : TStrings)
10100: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
10101: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
10102: Procedure WriteLn( const AOut : string)
10103: procedure Writeln(s: string);
10104: Procedure WriteLog( const FileName, LogLine : string)
10105: Procedure WriteRFCReply( AReply : TIdRFCReply)
10106: Procedure WriteRFCStrings( AStrings : TStrings)
10107: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
10108: Procedure WriteStream(AStream:TStream;const AAll:Boolean;const AWriteByteCount:Boolean;const ASize:Integer)
10109: Procedure WriteString( const Section, Ident, Value : String)
10110: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
10111: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
10112: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)');
10113: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)');
10114: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)');
10115: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
10116: procedure XMLSyntax1Click(Sender: TObject);
10117: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
10118: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
10119: Procedure ZeroFillStream( Stream : TMemoryStream)
10120: procedure XMLSyntax1Click(Sender: TObject);
10121: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
10122: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
10123: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
10124: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
10125: procedure(Sender, Source: TObject;X, Y: Integer)
10126: procedure(Sender, Target: TObject; X, Y: Integer)
10127: procedure(Sender: TObject; ASection, AWidth: Integer)
10128: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
10129: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
10130: procedure(Sender: TObject; var Action: TCloseAction)
10131: procedure(Sender: TObject; var CanClose: Boolean)
10132: procedure(Sender: TObject; var Key: Char);
10133: ProcedureName ProcedureNames
10134: ProcedureParametersCursor@
10135:
10136: *************Now Constructors constructor *************
10137:  Size is: 628 550 544 501 459 (381) (360) (270 246) (235)
10138:  Attach( VersionInfoData : Pointer; Size : Integer)
10139:  constructor Create( ABuckets : TBucketListSizes)
10140:  Create( ACallBackWnd : HWND)
10141:  Create( AClient : TCustomTaskDialog)
10142:  Create( AClient : TIdTelnet)
10143:  Create( ACollection : TCollection)
```

```
10144:  Create( ACollection : TFavoriteLinkItems)
10145:  Create( ACollection : TTaskDialogButtons)
10146:  Create( AConnection : TIdCustomHTTP)
10147:  Create( ACreateSuspended : Boolean)
10148:  Create( ADataSet : TCustomSQLDataSet)
10149:  CREATE( ADATASET : TDATASET)
10150:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
10151:  Create( AGrid : TCustomDBGrid)
10152:  Create( AGrid : TStringGrid; AIndex : Longint)
10153:  Create( AHTTP : TIdCustomHTTP)
10154:  Create( AListItems : TListItems)
10155:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
10156:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
10157:  Create( AOwner : TCommonCalendar)
10158:  Create( AOwner : TComponent)
10159:  CREATE( AOWNER : TCOMPONENT)
10160:  Create( AOwner : TCustomListView)
10161:  Create( AOwner : TCustomOutline)
10162:  Create( AOwner : TCustomRichEdit)
10163:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
10164:  Create( AOwner : TCustomTreeView)
10165:  Create( AOwner : TIdUserManager)
10166:  Create( AOwner : TListItems)
10167:  Create( AOwner : TObject; Handle : hDBICur; CBType : CBType; CBBuf : Pointer; CBBufSize : Integer;
       CallbackEvent : TBDECallbackEvent; Chain : Boolean)
10168:  CREATE( AOWNER : TPERSISTENT)
10169:  Create( AOwner : TPersistent)
10170:  Create( AOwner : TTable)
10171:  Create( AOwner : TTreeNodes)
10172:  Create( AOwner : TWinControl; const ClassName : string)
10173:  Create( AParent : TIdCustomHTTP)
10174:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
10175:  Create( AProvider : TBaseProvider)
10176:  Create( AProvider : TCustomProvider);
10177:  Create( AProvider : TDataSetProvider)
10178:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
10179:  Create( ASocket : TSocket)
10180:  Create( AStrings : TWideStrings)
10181:  Create( AToolBar : TToolBar)
10182:  Create( ATreeNodes : TTreeNodes)
10183:  Create( Autofill : boolean)
10184:  Create( AWebPageInfo : TAbstractWebPageInfo)
10185:  Create( AWebRequest : TWebRequest)
10186:  Create( Collection : TCollection)
10187:  Create( Collection : TIdMessageParts; ABody : TStrings)
10188:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
10189:  Create( Column : TColumn)
10190:  Create( const AConvFamily : TConvFamily; const ADescription : string)
10191:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
10192:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
10193:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
10194:  Create( const ATabSet : TTabSet)
10195:  Create( const Compensate : Boolean)
10196:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
10197:  Create( const FileName : string)
10198:  Create( const FileName : string; FileMode : Cardinal; const Name : string; Protect : Cardinal; const
       MaximumSize : Int64; const SecAttr : PSecurityAttributes);
10199:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
10200:  Create( const MaskValue : string)
10201:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
10202:  Create( const Prefix : string)
10203:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
10204:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
10205:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
10206:  Create( CoolBar : TCoolBar)
10207:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
10208:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
10209:  Create( DataSet :TDataSet; const Text: Widestring; Options : TFilterOptions; ParserOptions :
       TParserOptions; const FieldName : Widestring; DepFields : TBits; FieldMap : TFieldMap)
10210:  Create( DBCtrlGrid : TDBCtrlGrid)
10211:  Create( DSTableProducer : TDSTableProducer)
10212:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
10213:  Create( ErrorCode : DBIResult)
10214:  Create( Field : TBlobField; Mode : TBlobStreamMode)
10215:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
10216:  Create( HeaderControl : TCustomHeaderControl)
10217:  Create( HTTPRequest : TWebRequest)
10218:  Create( iStart : integer; sText : string)
10219:  Create( iValue : Integer)
10220:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
10221:  Create( MciErrNo : MCIERROR; const Msg : string)
10222:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
10223:  Create( Message : string; ErrorCode : DBResult)
10224:  Create( Msg : string)
10225:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
```

```
10226:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
10227:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
10228:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
10229:  Create(
       oSource:TniRegularExpressionState;oDestination:TniRegularExpressionState;xCharacters:TCharSet;bLambda:
       boolean)
10230:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
10231:  Create( Owner : TCustomComboBoxEx)
10232:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
10233:  Create( Owner : TPersistent)
10234:  Create( Params : TStrings)
10235:  Create( Size : Cardinal)
10236:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
10237:  Create( StatusBar : TCustomStatusBar)
10238:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
10239:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
10240:  Create(AHandle:Integer)
10241:  Create(AOwner: TComponent); virtual;
10242:  Create(const AURI : string)
10243:  Create(FileName:String;Mode:Word)
10244:  Create(Instance:THandle;ResName:String;ResType:PChar)
10245:  Create(Stream : TStream)
10246:  Create1( ADataset : TDataset);
10247:  Create1( const FileHandle : THandle; const Name : string; Protect : Cardinal; const MaximumSize : Int64;
       const SecAttr : PSecurityAttributes);
10248:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
10249:  Create2( Other : TObject);
10250:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
10251:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
10252:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
10253:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
10254:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
10255:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
10256:  CreateRes( Ident : Integer);
10257:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
10258:  CreateRes( ResStringRec : PResStringRec);
10259:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
10260:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
10261:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
10262:  CreateSize( AWidth, AHeight : Integer)
10263:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
10264:
10265: ------------------------------------------------------------------------------
10266: unit uPSI_MathMax;
10267: ------------------------------------------------------------------------------
10268:  CONSTS
10269:   Bernstein: Float = 0.280169499023869133036436491230 7;  // Bernstein constant
10270:   Cbrt2: Float     = 1.25992104989487316476721060727 82;  // CubeRoot(2)
10271:   Cbrt3: Float     = 1.44224957030740838232163831078 01;  // CubeRoot(3)
10272:   Cbrt10: Float    = 2.15443469003188372175929356651 94;  // CubeRoot(10)
10273:   Cbrt100: Float   = 4.64158883361277889241007635091 94;  // CubeRoot(100)
10274:   CbrtPi: Float    = 1.46459188756152326302014252726 38;  // CubeRoot(PI)
10275:   Catalan: Float   = 0.9159655941772190150546035149324;  // Catalan constant
10276:   PiJ:   Float       = 3.14159265358979323846264338327 95;  // PI
10277:   PI:  Extended =  3.14159265358979323846264338327 95);
10278:   PiOn2: Float     = 1.57079632679489661923132169163 98;  // PI / 2
10279:   PiOn3: Float     = 1.04719755119659774615421446109 32;  // PI / 3
10280:   PiOn4: Float     = 0.78539816339744830961566084581988; // PI / 4
10281:   Sqrt2: Float     = 1.41421356237309504880168872420 97;  // Sqrt(2)
10282:   Sqrt3: Float     = 1.73205080756887729352744634150 59;  // Sqrt(3)
10283:   Sqrt5: Float     = 2.23606797749978969640917366873 13;  // Sqrt(5)
10284:   Sqrt10: Float    = 3.16227766016837933199889354443 27;  // Sqrt(10)
10285:   SqrtPi: Float    = 1.77245385090551602729816748334 11;  // Sqrt(PI)
10286:   Sqrt2Pi: Float   = 2.50662827463100050241576528481 1;   // Sqrt(2 * PI)
10287:   TwoPi: Float     = 6.28318530717958647692528676655 9;   // 2 * PI
10288:   ThreePi: Float   = 9.42477796076937971538793014983 85;  // 3 * PI
10289:   Ln2: Float       = 0.69314718055994530941723212145818; // Ln(2)
10290:   Ln10: Float      = 2.30258509299404568401799145468 44;  // Ln(10)
10291:   LnPi: Float      = 1.14472988584940017414342735135 31;  // Ln(PI)
10292:   Log2J: Float     = 0.30102999566398119521373889472449; // Log10(2)
10293:   Log3: Float      = 0.47712125471966243729502790325512; // Log10(3)
10294:   LogPi: Float     = 0.49714987269413385435126828829 09;  // Log10(PI)
10295:   LogE: Float      = 0.43429448190325182765112891891661; // Log10(E)
10296:   E: Float         = 2.71828182845904523536028747135 27;  // Natural constant
10297:   hLn2Pi: Float    = 0.91893853320467274178032973640562; // Ln(2*PI)/2
10298:   inv2Pi: Float    = 0.15915494309189533576888376337251436203445964574046; // 0.5 / Pi
10299:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
10300:   GoldenMean: Float   = 1.61803398874989484820458683436563 8;  // GoldenMean
10301:   EulerMascheroni: Float = 0.57721566490153286060651209008 24;  // Euler GAMMA
10302:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
10303:   StDelta         : Extended = 0.00001;  {delta for difference equations}
10304:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
10305:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
10306:
10307:   MaxAngle',( 9223372036854775808.0);
10308:   MaxTanH',( 5678.2617031470719747459655389854);
```

```
10309:   MaxFactorial','LongInt').SetInt( 1754);
10310:   MaxFloatingPoint',(1.189731495357231765085759326628E+4932);
10311:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
10312:   MaxTanH',( 354.89135644669199842162284618659);
10313:   MaxFactorial','LongInt').SetInt( 170);
10314:   MaxFloatingPointD',(1.797693134862315907729305190789E+308);
10315:   MinFloatingPointD',(2.225073858507201383090232717332 4E-308);
10316:   MaxTanH',( 44.361419555836499802702855773323);
10317:   MaxFactorial','LongInt').SetInt( 33);
10318:   MaxFloatingPointS',( 3.402823669209384634633746074317 7E+38);
10319:   MinFloatingPointS',( 1.175494350822287507968736537222 2E-38);
10320:   PiExt',( 3.14159265358979323846264338327 95);
10321:   RatioDegToRad',( PiExt / 180.0);
10322:   RatioGradToRad',( PiExt / 200.0);
10323:   RatioDegToGrad',( 200.0 / 180.0);
10324:   RatioGradToDeg',( 180.0 / 200.0);
10325:
10326:   MinByte       = Low(Byte);
10327:   MaxByte       = High(Byte);
10328:   MinWord       = Low(Word);
10329:   MaxWord       = High(Word);
10330:   MinShortInt   = Low(ShortInt);
10331:   MaxShortInt   = High(ShortInt);
10332:   MinSmallInt   = Low(SmallInt);
10333:   MaxSmallInt   = High(SmallInt);
10334:   MinLongWord   = LongWord(Low(LongWord));
10335:   MaxLongWord   = LongWord(High(LongWord));
10336:   MinLongInt    = LongInt(Low(LongInt));
10337:   MaxLongInt    = LongInt(High(LongInt));
10338:   MinInt64      = Int64(Low(Int64));
10339:   MaxInt64      = Int64(High(Int64));
10340:   MinInteger    = Integer(Low(Integer));
10341:   MaxInteger    = Integer(High(Integer));
10342:   MinCardinal   = Cardinal(Low(Cardinal));
10343:   MaxCardinal   = Cardinal(High(Cardinal));
10344:   MinNativeUInt = NativeUInt(Low(NativeUInt));
10345:   MaxNativeUInt = NativeUInt(High(NativeUInt));
10346:   MinNativeInt  = NativeInt(Low(NativeInt));
10347:   MaxNativeInt  = NativeInt(High(NativeInt));
10348:
10349:   //*******************from DMath.Dll Lib of types.inc in source\dmath_dll
10350:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
10351:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
10352:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
10353:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
10354:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
10355:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
10356:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
10357:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
10358:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
10359:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
10360:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
10361:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
10362:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
10363:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
10364:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
10365:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
10366:   MaxLog = 709.7827128933840;
10367:   MinLog = -708.3964185322641;
10368:   MaxFac = 170;
10369:   MaxGam = 171.624376956302;
10370:   MaxLgm = 2.556348E+305;
10371:   SingleCompareDelta   = 1.0E-34;
10372:   DoubleCompareDelta   = 1.0E-280;
10373:   {$IFDEF CLR}
10374:   ExtendedCompareDelta = DoubleCompareDelta;
10375:   {$ELSE}
10376:   ExtendedCompareDelta = 1.0E-4400;
10377:   {$ENDIF}
10378:   Bytes1KB  = 1024;
10379:   Bytes1MB  = 1024 * Bytes1KB;
10380:   Bytes1GB  = 1024 * Bytes1MB;
10381:   Bytes64KB = 64 * Bytes1KB;
10382:   Bytes64MB = 64 * Bytes1MB;
10383:   Bytes2GB  = 2 * LongWord(Bytes1GB);
10384:     clBlack32', $FF000000 ));
10385:     clDimGray32', $FF3F3F3F ));
10386:     clGray32', $FF7F7F7F ));
10387:     clLightGray32', $FFBFBFBF ));
10388:     clWhite32', $FFFFFFFF ));
10389:     clMaroon32', $FF7F0000 ));
10390:     clGreen32', $FF007F00 ));
10391:     clOlive32', $FF7F7F00 ));
10392:     clNavy32', $FF00007F ));
10393:     clPurple32', $FF7F007F ));
10394:     clTeal32', $FF007F7F ));
```

```
10395:     clRed32', $FFFF0000 ));
10396:     clLime32', $FF00FF00 ));
10397:     clYellow32', $FFFFFF00 ));
10398:     clBlue32', $FF0000FF ));
10399:     clFuchsia32', $FFFF00FF ));
10400:     clAqua32', $FF00FFFF ));
10401:     clAliceBlue32', $FFF0F8FF ));
10402:     clAntiqueWhite32', $FFFAEBD7 ));
10403:     clAquamarine32', $FF7FFFD4 ));
10404:     clAzure32', $FFF0FFFF ));
10405:     clBeige32', $FFF5F5DC ));
10406:     clBisque32', $FFFFE4C4 ));
10407:     clBlancheDalmond32', $FFFFEBCD ));
10408:     clBlueViolet32', $FF8A2BE2 ));
10409:     clBrown32', $FFA52A2A ));
10410:     clBurlyWood32', $FFDEB887 ));
10411:     clCadetblue32', $FF5F9EA0 ));
10412:     clChartReuse32', $FF7FFF00 ));
10413:     clChocolate32', $FFD2691E ));
10414:     clCoral32', $FFFF7F50 ));
10415:     clCornFlowerBlue32', $FF6495ED ));
10416:     clCornSilk32', $FFFFF8DC ));
10417:     clCrimson32', $FFDC143C ));
10418:     clDarkBlue32', $FF00008B ));
10419:     clDarkCyan32', $FF008B8B ));
10420:     clDarkGoldenRod32', $FFB8860B ));
10421:     clDarkGray32', $FFA9A9A9 ));
10422:     clDarkGreen32', $FF006400 ));
10423:     clDarkGrey32', $FFA9A9A9 ));
10424:     clDarkKhaki32', $FFBDB76B ));
10425:     clDarkMagenta32', $FF8B008B ));
10426:     clDarkOliveGreen32', $FF556B2F ));
10427:     clDarkOrange32', $FFFF8C00 ));
10428:     clDarkOrchid32', $FF9932CC ));
10429:     clDarkRed32', $FF8B0000 ));
10430:     clDarkSalmon32', $FFE9967A ));
10431:     clDarkSeaGreen32', $FF8FBC8F ));
10432:     clDarkSlateBlue32', $FF483D8B ));
10433:     clDarkSlateGray32', $FF2F4F4F ));
10434:     clDarkSlateGrey32', $FF2F4F4F ));
10435:     clDarkTurquoise32', $FF00CED1 ));
10436:     clDarkViolet32', $FF9400D3 ));
10437:     clDeepPink32', $FFFF1493 ));
10438:     clDeepSkyBlue32', $FF00BFFF ));
10439:     clDodgerBlue32', $FF1E90FF ));
10440:     clFireBrick32', $FFB22222 ));
10441:     clFloralWhite32', $FFFFFAF0 ));
10442:     clGainsBoro32', $FFDCDCDC ));
10443:     clGhostWhite32', $FFF8F8FF ));
10444:     clGold32', $FFFFD700 ));
10445:     clGoldenRod32', $FFDAA520 ));
10446:     clGreenYellow32', $FFADFF2F ));
10447:     clGrey32', $FF808080 ));
10448:     clHoneyDew32', $FFF0FFF0 ));
10449:     clHotPink32', $FFFF69B4 ));
10450:     clIndianRed32', $FFCD5C5C ));
10451:     clIndigo32', $FF4B0082 ));
10452:     clIvory32', $FFFFFFF0 ));
10453:     clKhaki32', $FFF0E68C ));
10454:     clLavender32', $FFE6E6FA ));
10455:     clLavenderBlush32', $FFFFF0F5 ));
10456:     clLawnGreen32', $FF7CFC00 ));
10457:     clLemonChiffon32', $FFFFFACD ));
10458:     clLightBlue32', $FFADD8E6 ));
10459:     clLightCoral32', $FFF08080 ));
10460:     clLightCyan32', $FFE0FFFF ));
10461:     clLightGoldenRodYellow32', $FFFAFAD2 ));
10462:     clLightGreen32', $FF90EE90 ));
10463:     clLightGrey32', $FFD3D3D3 ));
10464:     clLightPink32', $FFFFB6C1 ));
10465:     clLightSalmon32', $FFFFA07A ));
10466:     clLightSeagreen32', $FF20B2AA ));
10467:     clLightSkyblue32', $FF87CEFA ));
10468:     clLightSlategray32', $FF778899 ));
10469:     clLightSlategrey32', $FF778899 ));
10470:     clLightSteelblue32', $FFB0C4DE ));
10471:     clLightYellow32', $FFFFFFE0 ));
10472:     clLtGray32', $FFC0C0C0 ));
10473:     clMedGray32', $FFA0A0A4 ));
10474:     clDkGray32', $FF808080 ));
10475:     clMoneyGreen32', $FFC0DCC0 ));
10476:     clLegacySkyBlue32', $FFA6CAF0 ));
10477:     clCream32', $FFFFFBF0 ));
10478:     clLimeGreen32', $FF32CD32 ));
10479:     clLinen32', $FFFAF0E6 ));
10480:     clMediumAquamarine32', $FF66CDAA ));
```

```
10481:     clMediumBlue32', $FF0000CD ));
10482:     clMediumOrchid32', $FFBA55D3 ));
10483:     clMediumPurple32', $FF9370DB ));
10484:     clMediumSeaGreen32', $FF3CB371 ));
10485:     clMediumSlateBlue32', $FF7B68EE ));
10486:     clMediumSpringGreen32', $FF00FA9A ));
10487:     clMediumTurquoise32', $FF48D1CC ));
10488:     clMediumVioletRed32', $FFC71585 ));
10489:     clMidnightBlue32', $FF191970 ));
10490:     clMintCream32', $FFF5FFFA ));
10491:     clMistyRose32', $FFFFE4E1 ));
10492:     clMoccasin32', $FFFFE4B5 ));
10493:     clNavajoWhite32', $FFFFDEAD ));
10494:     clOldLace32', $FFFDF5E6 ));
10495:     clOliveDrab32', $FF6B8E23 ));
10496:     clOrange32', $FFFFA500 ));
10497:     clOrangeRed32', $FFFF4500 ));
10498:     clOrchid32', $FFDA70D6 ));
10499:     clPaleGoldenRod32', $FFEEE8AA ));
10500:     clPaleGreen32', $FF98FB98 ));
10501:     clPaleTurquoise32', $FFAFEEEE ));
10502:     clPaleVioletred32', $FFDB7093 ));
10503:     clPapayaWhip32', $FFFFEFD5 ));
10504:     clPeachPuff32', $FFFFDAB9 ));
10505:     clPeru32', $FFCD853F ));
10506:     clPlum32', $FFDDA0DD ));
10507:     clPowderBlue32', $FFB0E0E6 ));
10508:     clRosyBrown32', $FFBC8F8F ));
10509:     clRoyalBlue32', $FF4169E1 ));
10510:     clSaddleBrown32', $FF8B4513 ));
10511:     clSalmon32', $FFFA8072 ));
10512:     clSandyBrown32', $FFF4A460 ));
10513:     clSeaGreen32', $FF2E8B57 ));
10514:     clSeaShell32', $FFFFF5EE ));
10515:     clSienna32', $FFA0522D ));
10516:     clSilver32', $FFC0C0C0 ));
10517:     clSkyblue32', $FF87CEEB ));
10518:     clSlateBlue32', $FF6A5ACD ));
10519:     clSlateGray32', $FF708090 ));
10520:     clSlateGrey32', $FF708090 ));
10521:     clSnow32', $FFFFFAFA ));
10522:     clSpringgreen32', $FF00FF7F ));
10523:     clSteelblue32', $FF4682B4 ));
10524:     clTan32', $FFD2B48C ));
10525:     clThistle32', $FFD8BFD8 ));
10526:     clTomato32', $FFFF6347 ));
10527:     clTurquoise32', $FF40E0D0 ));
10528:     clViolet32', $FFEE82EE ));
10529:     clWheat32', $FFF5DEB3 ));
10530:     clWhitesmoke32', $FFF5F5F5 ));
10531:     clYellowgreen32', $FF9ACD32 ));
10532:     clTrWhite32', $7FFFFFFF ));
10533:     clTrBlack32', $7F000000 ));
10534:     clTrRed32', $7FFF0000 ));
10535:     clTrGreen32', $7F00FF00 ));
10536:     clTrBlue32', $7F0000FF ));
10537:   // Fixed point math constants
10538:   FixedOne = $10000;
10539:   FixedHalf = $7FFF;
10540:   FixedPI  = Round(PI * FixedOne);
10541:   FixedToFloat = 1/FixedOne;
10542:
10543:  Special Types
10544: ****************************************************
10545:   type Complex = record
10546:     X, Y : Float;
10547:   end;
10548:   type TVector     = array of Float;
10549:   TIntVector  = array of Integer;
10550:   TCompVector = array of Complex;
10551:   TBoolVector = array of Boolean;
10552:   TStrVector  = array of String;
10553:   TMatrix     = array of TVector;
10554:   TIntMatrix  = array of TIntVector;
10555:   TCompMatrix = array of TCompVector;
10556:   TBoolMatrix = array of TBoolVector;
10557:   TStrMatrix  = array of TStrVector;
10558:   TByteArray = array[0..32767] of byte; !
10559:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
10560:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
10561:   T2StringArray = array of array of string;
10562:   T2IntegerArray = array of array of integer;
10563:   AddTypeS('INT_PTR', 'Integer');
10564:   AddTypeS('LONG_PTR', 'Integer');
10565:   AddTypeS('UINT_PTR', 'Cardinal');
10566:   AddTypeS('ULONG_PTR', 'Cardinal');
```

```
10567:   AddTypeS('DWORD_PTR', 'ULONG_PTR');
10568:   TIntegerDynArray', 'array of Integer');
10569:   TCardinalDynArray', 'array of Cardinal');
10570:   TWordDynArray', 'array of Word');
10571:   TSmallIntDynArray', 'array of SmallInt');
10572:   TByteDynArray', 'array of Byte');
10573:   TShortIntDynArray', 'array of ShortInt');
10574:   TInt64DynArray', 'array of Int64');
10575:   TLongWordDynArray', 'array of LongWord');
10576:   TSingleDynArray', 'array of Single');
10577:   TDoubleDynArray', 'array of Double');
10578:   TBooleanDynArray', 'array of Boolean');
10579:   TStringDynArray', 'array of string');
10580:   TWideStringDynArray', 'array of WideString');
10581:   TDynByteArray     = array of Byte;
10582:   TDynShortintArray = array of Shortint;
10583:   TDynSmallintArray = array of Smallint;
10584:   TDynWordArray     = array of Word;
10585:   TDynIntegerArray  = array of Integer;
10586:   TDynLongintArray  = array of Longint;
10587:   TDynCardinalArray = array of Cardinal;
10588:   TDynInt64Array    = array of Int64;
10589:   TDynExtendedArray = array of Extended;
10590:   TDynDoubleArray   = array of Double;
10591:   TDynSingleArray   = array of Single;
10592:   TDynFloatArray    = array of Float;
10593:   TDynPointerArray  = array of Pointer;
10594:   TDynStringArray   = array of string;
10595:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
10596:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
10597:   TSynSearchOptions = set of TSynSearchOption;
10598:
10599:
10600:
10601: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
10602: ----------------------------------------------------------------------------
10603: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
10604: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
10605: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
10606: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
10607: function CheckStringSum(vstring: string): integer;
10608: function HexToInt(HexNum: string): LongInt;
10609: function IntToBin(Int: Integer): String;
10610: function BinToInt(Binary: String): Integer;
10611: function HexToBin(HexNum: string): string; external2
10612: function BinToHex(Binary: String): string;
10613: function IntToFloat(i: Integer): double;
10614: function AddThousandSeparator(S: string; myChr: Char): string;
10615: function Max3(const X,Y,Z: Integer): Integer;
10616: procedure Swap(var X,Y: char); // faster without inline
10617: procedure ReverseString(var S: String);
10618: function CharToHexStr(Value: Char): string;
10619: function CharToUniCode(Value: Char): string;
10620: function Hex2Dec(Value: Str002): Byte;
10621: function HexStrCodeToStr(Value: string): string;
10622: function HexToStr(i: integer; value: string): string;
10623: function UniCodeToStr(Value: string): string;
10624: function CRC16(statement: string): string;
10625: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
10626: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
10627: procedure ShowInterfaces(myFile: string);
10628: function Fact2(av: integer): extended;
10629: Function BoolToStr(B: Boolean): string;
10630: Function GCD(x, y : LongInt) : LongInt;
10631: function LCM(m,n: longint): longint;
10632: function GetASCII: string;
10633: function GetItemHeight(Font: TFont): Integer;
10634: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
10635: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
10636: function getHINSTANCE: longword;
10637: function getHMODULE: longword;
10638: function GetASCII: string;
10639: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
10640: function WordIsOk(const AWord: string; var VW: Word): boolean;
10641: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
10642: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
10643: function SafeStr(const s: string): string;
10644: function ExtractUrlPath(const FileName: string): string;
10645: function ExtractUrlName(const FileName: string): string;
10646: function IsInternet: boolean;
10647: function RotateLeft1Bit_u32( Value: uint32): uint32;
10648: procedure LinearRegression(const KnownY: array of Double;const KnownX: array of Double;NData : Integer;
       var LF : TStLinEst; ErrorStats : Boolean);
10649: procedure getEnvironmentInfo;
10650: procedure AntiFreeze;
10651: function GetCPUSpeed: Double;
```

```
10652: function IsVirtualPcGuest : Boolean;
10653: function IsVmWareGuest : Boolean;
10654: procedure StartSerialDialog;
10655: function IsWoW64: boolean;
10656: function IsWow64String(var s: string): Boolean;
10657: procedure StartThreadDemo;
10658: Function RGB(R,G,B: Byte): TColor;
10659: Function Sendln(amess: string): boolean;
10660: Procedure maXbox;
10661: Function AspectRatio(aWidth, aHeight: Integer): String;
10662: function wget(aURL, afile: string): boolean;
10663: procedure PrintList(Value: TStringList);
10664: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
10665: procedure getEnvironmentInfo;
10666: procedure AntiFreeze;
10667: function getBitmap(apath: string): TBitmap;
10668: procedure ShowMessageBig(const aText : string);
10669: function YesNoDialog(const ACaption, AMsg: string): boolean;
10670: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
10671: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
10672: //function myStrToBytes(const Value: String): TBytes;
10673: //function myBytesToStr(const Value: TBytes): String;
10674: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
10675: function getBitmap(apath: string): TBitmap;
10676: procedure ShowMessageBig(const aText : string);
10677: Function StrToBytes(const Value: String): TBytes;
10678: Function BytesToStr(const Value: TBytes): String;
10679: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
10680: function ReverseDNSLookup(const IPAddress: String; const DNSServer:String; Timeout,Retries: Integer; var
       HostName: String): Boolean;
10681: function FindInPaths(const fileName, paths : String) : String;
10682: procedure initHexArray(var hexn: THexArray);
10683: function josephusG(n,k: integer; var graphout: string): integer;
10684: function isPowerof2(num: int64): boolean;
10685: function powerOf2(exponent: integer): int64;
10686: function getBigPI: string;
10687: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
10688:  function GetASCIILine: string;
10689: procedure MakeComplexSound(N:integer {stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
10690:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
10691: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
10692: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
10693: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
10694: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
10695:
10696:
10697: // News of 3.9.8
10698: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
10699: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
10700: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
10701: JvChart - TJvChart Component - 2009 Public
10702: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
10703: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
10704: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
10705: DMath DLL included incl. Demos
10706: Interface Navigator menu/View/Intf Navigator
10707: Unit Explorer menu/Debug/Units Explorer
10708: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
10709: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
10710: Script History to 9 Files WebServer light /Options/Addons/WebServer
10711: Full Text Finder, JVSimLogic Simulator Package
10712: Halt-Stop Program in Menu, WebServer2, Stop Event ,
10713: Conversion Routines, Prebuild Forms, CodeSearch
10714: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
10715: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
10716: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
10717: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
10718: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
10719: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
10720:
10721: add routines in 3.9.7.5
10722: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
10723: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
10724: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
10725: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
10726: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
10727: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
10728:
10729: ////////////////////////// TestUnits ////////////////////////////
10730:   SelftestPEM;
10731:   SelfTestCFundamentUtils;
10732:   SelfTestCFileUtils;
10733:   SelfTestCDateTime;
10734:   SelfTestCTimer;
10735:   SelfTestCRandom;
10736:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter');
```

```
10737:            Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath');
10738:
10739:   TGraphicControl = class(TControl)
10740:   private
10741:     FCanvas: TCanvas;
10742:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
10743:   protected
10744:     procedure Paint; virtual;
10745:     property Canvas: TCanvas read FCanvas;
10746:   public
10747:     constructor Create(AOwner: TComponent); override;
10748:     destructor Destroy; override;
10749:   end;
10750:
10751:   TCustomControl = class(TWinControl)
10752:   private
10753:     FCanvas: TCanvas;
10754:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
10755:   protected
10756:     procedure Paint; virtual;
10757:     procedure PaintWindow(DC: HDC); override;
10758:     property Canvas: TCanvas read FCanvas;
10759:   public
10760:     constructor Create(AOwner: TComponent); override;
10761:     destructor Destroy; override;
10762:   end;
10763:     RegisterPublishedProperties;
10764:     ('ONCHANGE', 'TNotifyEvent', iptrw);
10765:     ('ONCLICK', 'TNotifyEvent', iptrw);
10766:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
10767:     ('ONENTER', 'TNotifyEvent', iptrw);
10768:     ('ONEXIT', 'TNotifyEvent', iptrw);
10769:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
10770:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
10771:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
10772:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
10773:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
10774: //*************************************************************************
10775:   // To stop the while loop, click on Options/Show Include (boolean switch)!
10776:   Control a loop in a script with a form event:
10777:   IncludeON;   //control the while loop
10778:   while maxform1.ShowInclude1.checked do begin   //menu event Options/Show Include
10779:
10780: //-------------------------------------------------------------------------
10781: //**************mX4 ini-file Configuration***********************************
10782: //-------------------------------------------------------------------------
10783:   file maxboxdef.ini
10784:
10785: //*** Definitions for maXbox mX3 ***
10786: [FORM]
10787: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
10788: FONTSIZE=14
10789: EXTENSION=txt
10790: SCREENX=1386
10791: SCREENY=1077
10792: MEMHEIGHT=350
10793: PRINTFONT=Courier New //GUI Settings
10794: LINENUMBERS=Y   //line numbers at gutter in editor at left side
10795: EXCEPTIONLOG=Y  //store exceptions in log file - menu Debug/Show Last Exceptions
10796: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
10797: BOOTSCRIPT=Y    //enabling load a boot script
10798: MEMORYREPORT=Y  //shows memory report on closing maXbox
10799: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox\maxbox3\docs\
10800: NAVIGATOR=N     //shows function list at the right side of editor
10801: NAVWIDTH=350    //width of the right side interface list <CTRL L>
10802: [WEB]
10803: IPPORT=8080     //for internal webserver - menu /Options/Add Ons/WebServer
10804: IPHOST=192.168.1.53
10805: ROOTCERT=filepathY
10806: SCERT=filepathY
10807: RSAKEY=filepathY
10808: VERSIONCHECK=Y
10809:
10810: //-------------------------------------------------------------------------
10811: //**************mX4 maildef.ini ini-file Configuration********************
10812: //-------------------------------------------------------------------------
10813:
10814: //*** Definitions for maXMail ***
10815: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
10816: [MAXMAIL]
10817: HOST=getmail.softwareschule.ch
10818: USER=mailusername
10819: PASS=password
10820: PORT=110
10821: SSL=Y
10822: BODY=Y
```

```
10823: LAST=5
10824:
10825:
10826: //-------------------------------------------------------------------------
10827: //**************mX4 Macro Tags  ********************************************
10828: //-------------------------------------------------------- ----------------
10829:
10830:
10831:   asm #name, #date, #host, #path, #file, #head, #sign #tech end
10832:
10833: //Tag Macros
10834: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
10835: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
10836: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
10837: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
10838: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
10839: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
10840: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
10841: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
10842:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
10843: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
10844: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
10845:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
10846: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
10847:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
10848: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
10849:
10850: //Replace Macros
10851:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
10852:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
10853:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
10854:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
10855:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
10856:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
10857:
10858:
10859:
10860:
10861: //-------------------------------------------------------------------------
10862: //**************mX4 Public  Tools API ****************************************
10863: //-------------------------------------------------------------------------
10864:   file : unit uPSI_fMain.pas;              OTAP Open Tools API Catalog
10865:  // Those functions concern the editor and preprocessor, all of the IDE
10866:  Example: Call it with maxform1.Info1Click(self)
10867:  Note: Call all Methods with maxForm1., e.g.:
10868:                       maxForm1.ShellStyle1Click(self);
10869:
10870: procedure SIRegister_fMain(CL: TPSPascalCompiler);
10871: begin
10872:  Const('BYTECODE','String').SetString( 'bytecode.txt');
10873:  Const('PSTEXT','String').SetString( 'PS Scriptfiles (*.txt)|*.TXT');
10874:  Const('PSMODEL','String').SetString( 'PS Modelfiles (*.uc)|*.UC');
10875:  Const('PSPASCAL','String').SetString( 'PS Pascalfiles (*.pas)|*.PAS');
10876:  Const('PSINC','String').SetString( 'PS Includes (*.inc)|*.INC');
10877:  Const('DEFFILENAME','String').SetString( 'firstdemo.txt');
10878:  Const('DEFINIFILE','String').SetString( 'maxboxdef.ini');
10879:  Const('EXCEPTLOGFILE','String').SetString( 'maxboxerrorlog.txt');
10880:  Const('ALLFUNCTIONSLIST','String').SetString( 'upsi_allfunctionslist.txt');
10881:  Const('ALLFUNCTIONSLISTPDF','String').SetString( 'maxbox_functions_all.pdf');
10882:  Const('ALLOBJECTSLIST','String').SetString( 'docs\VCL.pdf');
10883:  Const('ALLRESOURCELIST','String').SetString( 'docs\upsi_allresourcelist.txt');
10884:  Const('INCLUDEBOX','String').SetString( 'pas_includebox.inc');
10885:  Const('BOOTSCRIPT','String').SetString( 'maxbootscript.txt');
10886:  Const('MBVERSION','String').SetString( '3.9.9.5');
10887:  Const('MBVER','String').SetString( '399');
10888:  Const('EXENAME','String').SetString( 'maXbox3.exe');
10889:  Const('MXSITE','String').SetString( 'http://www.softwareschule.ch/maxbox.htm');
10890:  Const('MXVERSIONFILE','String').SetString( 'http://www.softwareschule.ch/maxvfile.txt');
10891:  Const('MXINTERNETCHECK','String').SetString( 'www.ask.com');
10892:  Const('MXMAIL','String').SetString( 'max@kleiner.com');
10893:  Const('TAB','Char').SetString( #$09);
10894:  Const('CODECOMPLETION','String').SetString( 'bds_delphi.dci');
10895:  SIRegister_TMaxForm1(CL);
10896: end;
10897:
10898:   with CL.AddClassN(CL.FindClass('TForm'),'TMaxForm1') do begin
10899:     memo2', 'TMemo', iptrw);
10900:     memo1', 'TSynMemo', iptrw);
10901:     CB1SCList', 'TComboBox', iptrw);
10902:     mxNavigator', 'TComboBox', iptrw);
10903:     IPHost', 'string', iptrw);
10904:     IPPort', 'integer', iptrw);
10905:     COMPort', 'integer', iptrw);      //3.9.6.4
10906:     Splitter1', 'TSplitter', iptrw);
10907:     PSScript', 'TPSScript', iptrw);
10908:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
```

```
10909:     MainMenu1', 'TMainMenu', iptrw);
10910:     Program1', 'TMenuItem', iptrw);
10911:     Compile1', 'TMenuItem', iptrw);
10912:     Files1', 'TMenuItem', iptrw);
10913:     open1', 'TMenuItem', iptrw);
10914:     Save2', 'TMenuItem', iptrw);
10915:     Options1', 'TMenuItem', iptrw);
10916:     Savebefore1', 'TMenuItem', iptrw);
10917:     Largefont1', 'TMenuItem', iptrw);
10918:     sBytecode1', 'TMenuItem', iptrw);
10919:     Saveas3', 'TMenuItem', iptrw);
10920:     Clear1', 'TMenuItem', iptrw);
10921:     Slinenumbers1', 'TMenuItem', iptrw);
10922:     About1', 'TMenuItem', iptrw);
10923:     Search1', 'TMenuItem', iptrw);
10924:     SynPasSyn1', 'TSynPasSyn', iptrw);
10925:     memo1', 'TSynMemo', iptrw);
10926:     SynEditSearch1', 'TSynEditSearch', iptrw);
10927:     WordWrap1', 'TMenuItem', iptrw);
10928:     XPManifest1', 'TXPManifest', iptrw);
10929:     SearchNext1', 'TMenuItem', iptrw);
10930:     Replace1', 'TMenuItem', iptrw);
10931:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
10932:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
10933:     ShowInclude1', 'TMenuItem', iptrw);
10934:     SynEditPrint1', 'TSynEditPrint', iptrw);
10935:     Printout1', 'TMenuItem', iptrw);
10936:     mnPrintColors1', 'TMenuItem', iptrw);
10937:     dlgFilePrint', 'TPrintDialog', iptrw);
10938:     dlgPrintFont1', 'TFontDialog', iptrw);
10939:     mnuPrintFont1', 'TMenuItem', iptrw);
10940:     Include1', 'TMenuItem', iptrw);
10941:     CodeCompletionList1', 'TMenuItem', iptrw);
10942:     IncludeList1', 'TMenuItem', iptrw);
10943:     ImageList1', 'TImageList', iptrw);
10944:     ImageList2', 'TImageList', iptrw);
10945:     CoolBar1', 'TCoolBar', iptrw);
10946:     ToolBar1', 'TToolBar', iptrw);
10947:     tbtnLoad', 'TToolButton', iptrw);
10948:     ToolButton2', 'TToolButton', iptrw);
10949:     tbtnFind', 'TToolButton', iptrw);
10950:     tbtnCompile', 'TToolButton', iptrw);
10951:     tbtnTrans', 'TToolButton', iptrw);
10952:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
10953:     toolbtnTutorial', 'TToolButton', iptrw);
10954:     tbtn6res', 'TToolButton', iptrw);
10955:     ToolButton5', 'TToolButton', iptrw);
10956:     ToolButton1', 'TToolButton', iptrw);
10957:     ToolButton3', 'TToolButton', iptrw);
10958:     statusBar1', 'TStatusBar', iptrw);
10959:     SaveOutput1', 'TMenuItem', iptrw);
10960:     ExportClipboard1', 'TMenuItem', iptrw);
10961:     Close1', 'TMenuItem', iptrw);
10962:     Manual1', 'TMenuItem', iptrw);
10963:     About2', 'TMenuItem', iptrw);
10964:     loadLastfile1', 'TMenuItem', iptrw);
10965:     imglogo', 'TImage', iptrw);
10966:     cedebug', 'TPSScriptDebugger', iptrw);
10967:     debugPopupMenu1', 'TPopupMenu', iptrw);
10968:     BreakPointMenu', 'TMenuItem', iptrw);
10969:     Decompile1', 'TMenuItem', iptrw);
10970:     StepInto1', 'TMenuItem', iptrw);
10971:     StepOut1', 'TMenuItem', iptrw);
10972:     Reset1', 'TMenuItem', iptrw);
10973:     DebugRun1', 'TMenuItem', iptrw);
10974:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
10975:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
10976:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
10977:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
10978:     tutorial4', 'TMenuItem', iptrw);
10979:     ExporttoClipboard1', 'TMenuItem', iptrw);
10980:     ImportfromClipboard1', 'TMenuItem', iptrw);
10981:     N4', 'TMenuItem', iptrw);
10982:     N5', 'TMenuItem', iptrw);
10983:     N6', 'TMenuItem', iptrw);
10984:     ImportfromClipboard2', 'TMenuItem', iptrw);
10985:     tutorial1', 'TMenuItem', iptrw);
10986:     N7', 'TMenuItem', iptrw);
10987:     ShowSpecChars1', 'TMenuItem', iptrw);
10988:     OpenDirectory1', 'TMenuItem', iptrw);
10989:     procMess', 'TMenuItem', iptrw);
10990:     tbtnUseCase', 'TToolButton', iptrw);
10991:     ToolButton7', 'TToolButton', iptrw);
10992:     EditFont1', 'TMenuItem', iptrw);
10993:     UseCase1', 'TMenuItem', iptrw);
10994:     tutorial21', 'TMenuItem', iptrw);
```

```
10995:     OpenUseCase1', 'TMenuItem', iptrw);
10996:     PSImport_DB1', 'TPSImport_DB', iptrw);
10997:     tutorial31', 'TMenuItem', iptrw);
10998:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
10999:     HTMLSyntax1', 'TMenuItem', iptrw);
11000:     ShowInterfaces1', 'TMenuItem', iptrw);
11001:     Tutorial5', 'TMenuItem', iptrw);
11002:     AllFunctionsList1', 'TMenuItem', iptrw);
11003:     ShowLastException1', 'TMenuItem', iptrw);
11004:     PlayMP31', 'TMenuItem', iptrw);
11005:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
11006:     texSyntax1', 'TMenuItem', iptrw);
11007:     N8', 'TMenuItem', iptrw);
11008:     GetEMails1', 'TMenuItem', iptrw);
11009:     SynCppSyn1', 'TSynCppSyn', iptrw);
11010:     CSyntax1', 'TMenuItem', iptrw);
11011:     Tutorial6', 'TMenuItem', iptrw);
11012:     New1', 'TMenuItem', iptrw);
11013:     AllObjectsList1', 'TMenuItem', iptrw);
11014:     LoadBytecode1', 'TMenuItem', iptrw);
11015:     CipherFile1', 'TMenuItem', iptrw);
11016:     N9', 'TMenuItem', iptrw);
11017:     N10', 'TMenuItem', iptrw);
11018:     Tutorial11', 'TMenuItem', iptrw);
11019:     Tutorial71', 'TMenuItem', iptrw);
11020:     UpdateService1', 'TMenuItem', iptrw);
11021:     PascalSchool1', 'TMenuItem', iptrw);
11022:     Tutorial81', 'TMenuItem', iptrw);
11023:     DelphiSite1', 'TMenuItem', iptrw);
11024:     Output1', 'TMenuItem', iptrw);
11025:     TerminalStyle1', 'TMenuItem', iptrw);
11026:     ReadOnly1', 'TMenuItem', iptrw);
11027:     ShellStyle1', 'TMenuItem', iptrw);
11028:     BigScreen1', 'TMenuItem', iptrw);
11029:     Tutorial91', 'TMenuItem', iptrw);
11030:     SaveOutput2', 'TMenuItem', iptrw);
11031:     N11', 'TMenuItem', iptrw);
11032:     SaveScreenshot', 'TMenuItem', iptrw);
11033:     Tutorial101', 'TMenuItem', iptrw);
11034:     SQLSyntax1', 'TMenuItem', iptrw);
11035:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
11036:     Console1', 'TMenuItem', iptrw);
11037:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
11038:     XMLSyntax1', 'TMenuItem', iptrw);
11039:     ComponentCount1', 'TMenuItem', iptrw);
11040:     NewInstance1', 'TMenuItem', iptrw);
11041:     toolbtnTutorial', 'TToolButton', iptrw);
11042:     Memory1', 'TMenuItem', iptrw);
11043:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
11044:     JavaSyntax1', 'TMenuItem', iptrw);
11045:     SyntaxCheck1', 'TMenuItem', iptrw);
11046:     Tutorial10Statistics1', 'TMenuItem', iptrw);
11047:     ScriptExplorer1', 'TMenuItem', iptrw);
11048:     FormOutput1', 'TMenuItem', iptrw);
11049:     ArduinoDump1', 'TMenuItem', iptrw);
11050:     AndroidDump1', 'TMenuItem', iptrw);
11051:     GotoEnd1', 'TMenuItem', iptrw);
11052:     AllResourceList1', 'TMenuItem', iptrw);
11053:     ToolButton4', 'TToolButton', iptrw);
11054:     tbtn6res', 'TToolButton', iptrw);
11055:     Tutorial11Forms1', 'TMenuItem', iptrw);
11056:     Tutorial12SQL1', 'TMenuItem', iptrw);
11057:     ResourceExplore1', 'TMenuItem', iptrw);
11058:     Info1', 'TMenuItem', iptrw);
11059:     N12', 'TMenuItem', iptrw);
11060:     CryptoBox1', 'TMenuItem', iptrw);
11061:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
11062:     CipherFile2', 'TMenuItem', iptrw);
11063:     N13', 'TMenuItem', iptrw);
11064:     ModulesCount1', 'TMenuItem', iptrw);
11065:     AddOns2', 'TMenuItem', iptrw);
11066:     N4GewinntGame1', 'TMenuItem', iptrw);
11067:     DocuforAddOns1', 'TMenuItem', iptrw);
11068:     Tutorial14Async1', 'TMenuItem', iptrw);
11069:     Lessons15Review1', 'TMenuItem', iptrw);
11070:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
11071:     PHPSyntax1', 'TMenuItem', iptrw);
11072:     Breakpoint1', 'TMenuItem', iptrw);
11073:     SerialRS2321', 'TMenuItem', iptrw);
11074:     N14', 'TMenuItem', iptrw);
11075:     SynCSSyn1', 'TSynCSSyn', iptrw);
11076:     CSyntax2', 'TMenuItem', iptrw);
11077:     Calculator1', 'TMenuItem', iptrw);
11078:     tbtnSerial', 'TToolButton', iptrw);
11079:     ToolButton8', 'TToolButton', iptrw);
11080:     Tutorial151', 'TMenuItem', iptrw);
```

```
11081:     N15', 'TMenuItem', iptrw);
11082:     N16', 'TMenuItem', iptrw);
11083:     ControlBar1', 'TControlBar', iptrw);
11084:     ToolBar2', 'TToolBar', iptrw);
11085:     BtnOpen', 'TToolButton', iptrw);
11086:     BtnSave', 'TToolButton', iptrw);
11087:     BtnPrint', 'TToolButton', iptrw);
11088:     BtnColors', 'TToolButton', iptrw);
11089:     btnClassReport', 'TToolButton', iptrw);
11090:     BtnRotateRight', 'TToolButton', iptrw);
11091:     BtnFullSize', 'TToolButton', iptrw);
11092:     BtnFitToWindowSize', 'TToolButton', iptrw);
11093:     BtnZoomMinus', 'TToolButton', iptrw);
11094:     BtnZoomPlus', 'TToolButton', iptrw);
11095:     Panel1', 'TPanel', iptrw);
11096:     LabelBrettgroesse', 'TLabel', iptrw);
11097:     CB1SCList', 'TComboBox', iptrw);
11098:     ImageListNormal', 'TImageList', iptrw);
11099:     spbtnexplore', 'TSpeedButton', iptrw);
11100:     spbtnexample', 'TSpeedButton', iptrw);
11101:     spbsaveas', 'TSpeedButton', iptrw);
11102:     imglogobox', 'TImage', iptrw);
11103:     EnlargeFont1', 'TMenuItem', iptrw);
11104:     EnlargeFont2', 'TMenuItem', iptrw);
11105:     ShrinkFont1', 'TMenuItem', iptrw);
11106:     ThreadDemo1', 'TMenuItem', iptrw);
11107:     HEXEditor1', 'TMenuItem', iptrw);
11108:     HEXView1', 'TMenuItem', iptrw);
11109:     HEXInspect1', 'TMenuItem', iptrw);
11110:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
11111:     ExporttoHTML1', 'TMenuItem', iptrw);
11112:     ClassCount1', 'TMenuItem', iptrw);
11113:     HTMLOutput1', 'TMenuItem', iptrw);
11114:     HEXEditor2', 'TMenuItem', iptrw);
11115:     Minesweeper1', 'TMenuItem', iptrw);
11116:     N17', 'TMenuItem', iptrw);
11117:     PicturePuzzle1', 'TMenuItem', iptrw);
11118:     sbvclhelp', 'TSpeedButton', iptrw);
11119:     DependencyWalker1', 'TMenuItem', iptrw);
11120:     WebScanner1', 'TMenuItem', iptrw);
11121:     View1', 'TMenuItem', iptrw);
11122:     mnToolbar1', 'TMenuItem', iptrw);
11123:     mnStatusbar2', 'TMenuItem', iptrw);
11124:     mnConsole2', 'TMenuItem', iptrw);
11125:     mnCoolbar2', 'TMenuItem', iptrw);
11126:     mnSplitter2', 'TMenuItem', iptrw);
11127:     WebServer1', 'TMenuItem', iptrw);
11128:     Tutorial17Server1', 'TMenuItem', iptrw);
11129:     Tutorial18Arduino1', 'TMenuItem', iptrw);
11130:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
11131:     PerlSyntax1', 'TMenuItem', iptrw);
11132:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
11133:     PythonSyntax1', 'TMenuItem', iptrw);
11134:     DMathLibrary1', 'TMenuItem', iptrw);
11135:     IntfNavigator1', 'TMenuItem', iptrw);
11136:     EnlargeFontConsole1', 'TMenuItem', iptrw);
11137:     ShrinkFontConsole1', 'TMenuItem', iptrw);
11138:     SetInterfaceList1', 'TMenuItem', iptrw);
11139:     popintfList', 'TPopupMenu', iptrw);
11140:     intfAdd1', 'TMenuItem', iptrw);
11141:     intfDelete1', 'TMenuItem', iptrw);
11142:     intfRefactor1', 'TMenuItem', iptrw);
11143:     Defactor1', 'TMenuItem', iptrw);
11144:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
11145:     Tutorial20Regex', 'TMenuItem', iptrw);
11146:     N18', 'TMenuItem', iptrw);
11147:     ManualE1', 'TMenuItem', iptrw);
11148:     FullTextFinder1', 'TMenuItem', iptrw);
11149:     Move1', 'TMenuItem', iptrw);
11150:     FractalDemo1', 'TMenuItem', iptrw);
11151:     Tutorial21Android1', 'TMenuItem', iptrw);
11152:     Tutorial0Function1', 'TMenuItem', iptrw);
11153:     SimuLogBox1', 'TMenuItem', iptrw);
11154:     OpenExamples1', 'TMenuItem', iptrw);
11155:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
11156:     JavaScriptSyntax1', 'TMenuItem', iptrw);
11157:     Halt1', 'TMenuItem', iptrw);
11158:     CodeSearch1', 'TMenuItem', iptrw);
11159:     SynRubySyn1', 'TSynRubySyn', iptrw);
11160:     RubySyntax1', 'TMenuItem', iptrw);
11161:     Undo1', 'TMenuItem', iptrw);
11162:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
11163:     LinuxShellScript1', 'TMenuItem', iptrw);
11164:     Rename1', 'TMenuItem', iptrw);
11165:     spdcodesearch', 'TSpeedButton', iptrw);
11166:     Preview1', 'TMenuItem', iptrw);
```

```
11167:     Tutorial22Services1', 'TMenuItem', iptrw);
11168:     Tutorial23RealTime1', 'TMenuItem', iptrw);
11169:     Configuration1', 'TMenuItem', iptrw);
11170:     MP3Player1', 'TMenuItem', iptrw);
11171:     DLLSpy1', 'TMenuItem', iptrw);
11172:     SynURIOpener1', 'TSynURIOpener', iptrw);
11173:     SynURISyn1', 'TSynURISyn', iptrw);
11174:     URILinksClicks1', 'TMenuItem', iptrw);
11175:     EditReplace1', 'TMenuItem', iptrw);
11176:     GotoLine1', 'TMenuItem', iptrw);
11177:     ActiveLineColor1', 'TMenuItem', iptrw);
11178:     ConfigFile1', 'TMenuItem', iptrw);
11179:     Sort1Intflist', 'TMenuItem', iptrw);
11180:     Redo1', 'TMenuItem', iptrw);
11181:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
11182:     Tutorial25Configuration1', 'TMenuItem', iptrw);
11183:     IndentSelection1', 'TMenuItem', iptrw);
11184:     UnindentSection1', 'TMenuItem', iptrw);
11185:     SkyStyle1', 'TMenuItem', iptrw);
11186:     N19', 'TMenuItem', iptrw);
11187:     CountWords1', 'TMenuItem', iptrw);
11188:     imbookmarkimages', 'TImageList', iptrw);
11189:     Bookmark11', 'TMenuItem', iptrw);
11190:     N20', 'TMenuItem', iptrw);
11191:     Bookmark21', 'TMenuItem', iptrw);
11192:     Bookmark31', 'TMenuItem', iptrw);
11193:     Bookmark41', 'TMenuItem', iptrw);
11194:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
11195:
11196:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)');
11197:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
11198:     Procedure PSScriptCompile( Sender : TPSScript)');
11199:     Procedure Compile1Click( Sender : TObject)');
11200:     Procedure PSScriptExecute( Sender : TPSScript)');
11201:     Procedure open1Click( Sender : TObject)');
11202:     Procedure Save2Click( Sender : TObject)');
11203:     Procedure Savebefore1Click( Sender : TObject)');
11204:     Procedure Largefont1Click( Sender : TObject)');
11205:     Procedure FormActivate( Sender : TObject)');
11206:     Procedure SBytecode1Click( Sender : TObject)');
11207:     Procedure FormKeyPress( Sender : TObject; var Key : Char)');
11208:     Procedure Saveas3Click( Sender : TObject)');
11209:     Procedure Clear1Click( Sender : TObject)');
11210:     Procedure Slinenumbers1Click( Sender : TObject)');
11211:     Procedure About1Click( Sender : TObject)');
11212:     Procedure Search1Click( Sender : TObject)');
11213:     Procedure FormCreate( Sender : TObject)');
11214:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
11215:                                                 var Action : TSynReplaceAction)');
11216:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)');
11217:     Procedure WordWrap1Click( Sender : TObject)');
11218:     Procedure SearchNext1Click( Sender : TObject)');
11219:     Procedure Replace1Click( Sender : TObject)');
11220:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,
       Output:String):Boolean;
11221:     Procedure ShowInclude1Click( Sender : TObject)');
11222:     Procedure Printout1Click( Sender : TObject)');
11223:     Procedure mnuPrintFont1Click( Sender : TObject)');
11224:     Procedure Include1Click( Sender : TObject)');
11225:     Procedure FormDestroy( Sender : TObject)');
11226:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)');
11227:     Procedure UpdateView1Click( Sender : TObject)');
11228:     Procedure CodeCompletionList1Click( Sender : TObject)');
11229:     Procedure SaveOutput1Click( Sender : TObject)');
11230:     Procedure ExportClipboard1Click( Sender : TObject)');
11231:     Procedure Close1Click( Sender : TObject)');
11232:     Procedure Manual1Click( Sender : TObject)');
11233:     Procedure LoadLastFile1Click( Sender : TObject)');
11234:     Procedure Memo1Change( Sender : TObject)');
11235:     Procedure Decompile1Click( Sender : TObject)');
11236:     Procedure StepInto1Click( Sender : TObject)');
11237:     Procedure StepOut1Click( Sender : TObject)');
11238:     Procedure Reset1Click( Sender : TObject)');
11239:     Procedure cedebugAfterExecute( Sender : TPSScript)');
11240:     Procedure cedebugBreakpoint(Sender: TObject; const FileName: String; Position,Row, Col: Cardinal)');
11241:     Procedure cedebugCompile( Sender : TPSScript)');
11242:     Procedure cedebugExecute( Sender : TPSScript)');
11243:     Procedure cedebugIdle( Sender : TObject)');
11244:     Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)');
11245:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Integer; var Special:Boolean;var FG,BG:TColor);
11246:     Procedure BreakPointMenuClick( Sender : TObject)');
11247:     Procedure DebugRun1Click( Sender : TObject)');
11248:     Procedure tutorial4Click( Sender : TObject)');
11249:     Procedure ImportfromClipboard1Click( Sender : TObject)');
11250:     Procedure ImportfromClipboard2Click( Sender : TObject)');
11251:     Procedure tutorial1Click( Sender : TObject)');
```

```
11252:     Procedure ShowSpecChars1Click( Sender : TObject)');
11253:     Procedure StatusBar1DblClick( Sender : TObject)');
11254:     Procedure PSScriptLine( Sender : TObject)');
11255:     Procedure OpenDirectory1Click( Sender : TObject)');
11256:     Procedure procMessClick( Sender : TObject)');
11257:     Procedure tbtnUseCaseClick( Sender : TObject)');
11258:     Procedure EditFont1Click( Sender : TObject)');
11259:     Procedure tutorial21Click( Sender : TObject)');
11260:     Procedure tutorial31Click( Sender : TObject)');
11261:     Procedure HTMLSyntax1Click( Sender : TObject)');
11262:     Procedure ShowInterfaces1Click( Sender : TObject)');
11263:     Procedure Tutorial5Click( Sender : TObject)');
11264:     Procedure ShowLastException1Click( Sender : TObject)');
11265:     Procedure PlayMP31Click( Sender : TObject)');
11266:     Procedure AllFunctionsList1Click( Sender : TObject)');
11267:     Procedure texSyntax1Click( Sender : TObject)');
11268:     Procedure GetEMails1Click( Sender : TObject)');
11269:     procedure DelphiSite1Click(Sender: TObject);');
11270:     procedure TerminalStyle1Click(Sender: TObject);');
11271:     procedure ReadOnly1Click(Sender: TObject);');
11272:     procedure ShellStyle1Click(Sender: TObject);');
11273:     procedure Console1Click(Sender: TObject);');     //3.2
11274:     procedure BigScreen1Click(Sender: TObject);');
11275:     procedure Tutorial91Click(Sender: TObject);');
11276:     procedure SaveScreenshotClick(Sender: TObject);');
11277:     procedure Tutorial101Click(Sender: TObject);');
11278:     procedure SQLSyntax1Click(Sender: TObject);');
11279:     procedure XMLSyntax1Click(Sender: TObject);');
11280:     procedure ComponentCount1Click(Sender: TObject);');
11281:     procedure NewInstance1Click(Sender: TObject);');
11282:     procedure CSyntax1Click(Sender: TObject);');
11283:     procedure Tutorial6Click(Sender: TObject);');
11284:     procedure New1Click(Sender: TObject);');
11285:     procedure AllObjectsList1Click(Sender: TObject);');
11286:     procedure LoadBytecode1Click(Sender: TObject);');
11287:     procedure CipherFile1Click(Sender: TObject);');  //V3.5
11288:     procedure NewInstance1Click(Sender: TObject);');
11289:     procedure toolbtnTutorialClick(Sender: TObject);');
11290:     procedure Memory1Click(Sender: TObject);');
11291:     procedure JavaSyntax1Click(Sender: TObject);');
11292:     procedure SyntaxCheck1Click(Sender: TObject);');
11293:     procedure ScriptExplorer1Click(Sender: TObject);');
11294:     procedure FormOutput1Click(Sender: TObject);');  //V3.6
11295:     procedure GotoEnd1Click(Sender: TObject);');
11296:     procedure AllResourceList1Click(Sender: TObject);');
11297:     procedure tbtn6resClick(Sender: TObject);');   //V3.7
11298:     procedure Info1Click(Sender: TObject);');
11299:     procedure Tutorial10Statistics1Click(Sender: TObject);');
11300:     procedure Tutorial11Forms1Click(Sender: TObject);');
11301:     procedure Tutorial12SQL1Click(Sender: TObject);');  //V3.8
11302:     procedure ResourceExplore1Click(Sender: TObject);');
11303:     procedure Info1Click(Sender: TObject);');
11304:     procedure CryptoBox1Click(Sender: TObject);');
11305:     procedure ModulesCount1Click(Sender: TObject);');
11306:     procedure N4GewinntGame1Click(Sender: TObject);');
11307:     procedure PHPSyntax1Click(Sender: TObject);');
11308:     procedure SerialRS2321Click(Sender: TObject);');
11309:     procedure CSyntax2Click(Sender: TObject);');
11310:     procedure Calculator1Click(Sender: TObject);');
11311:     procedure Tutorial13Ciphering1Click(Sender: TObject);');
11312:     procedure Tutorial14Async1Click(Sender: TObject);');
11313:     procedure PHPSyntax1Click(Sender: TObject);');
11314:     procedure BtnZoomPlusClick(Sender: TObject);');
11315:     procedure BtnZoomMinusClick(Sender: TObject);');
11316:     procedure btnClassReportClick(Sender: TObject);');
11317:     procedure ThreadDemo1Click(Sender: TObject);');
11318:     procedure HEXView1Click(Sender: TObject);');
11319:     procedure ExporttoHTML1Click(Sender: TObject);');
11320:     procedure Minesweeper1Click(Sender: TObject);');
11321:     procedure PicturePuzzle1Click(Sender: TObject);'); //V3.9
11322:     procedure sbvclhelpClick(Sender: TObject);');
11323:     procedure DependencyWalker1Click(Sender: TObject);');
11324:     procedure CB1SCListDrawItem(Control:TWinControl;Index: Integer;aRect:TRect;State:TOwnerDrawState);');
11325:     procedure WebScanner1Click(Sender: TObject);');
11326:     procedure mnToolbar1Click(Sender: TObject);');
11327:     procedure mnStatusbar2Click(Sender: TObject);');
11328:     procedure mnConsole2Click(Sender: TObject);');
11329:     procedure mnCoolbar2Click(Sender: TObject);');
11330:     procedure mnSplitter2Click(Sender: TObject);');
11331:     procedure WebServer1Click(Sender: TObject);');
11332:     procedure PerlSyntax1Click(Sender: TObject);');
11333:     procedure PythonSyntax1Click(Sender: TObject);');
11334:     procedure DMathLibrary1Click(Sender: TObject);');
11335:     procedure IntfNavigator1Click(Sender: TObject);');
11336:     procedure FullTextFinder1Click(Sender: TObject);');
11337:     function AppName: string;');
```

```
11338:     function ScriptName: string;');
11339:     function LastName: string;');
11340:     procedure FractalDemo1Click(Sender: TObject);');
11341:     procedure SimuLogBox1Click(Sender: TObject);');
11342:     procedure OpenExamples1Click(Sender: TObject);');
11343:     procedure Halt1Click(Sender: TObject);');
11344:     procedure Stop;');
11345:     procedure CodeSearch1Click(Sender: TObject);');
11346:     procedure RubySyntax1Click(Sender: TObject);');
11347:     procedure Undo1Click(Sender: TObject);');
11348:     procedure LinuxShellScript1Click(Sender: TObject);');
11349:     procedure WebScannerDirect(urls: string);');
11350:     procedure WebScanner(urls: string);');
11351:     procedure LoadInterfaceList2;');
11352:     procedure DLLSpy1Click(Sender: TObject);');
11353:     procedure Memo1DblClick(Sender: TObject);');
11354:     procedure URILinksClicks1Click(Sender: TObject);');
11355:     procedure GotoLine1Click(Sender: TObject);');
11356:     procedure ConfigFile1Click(Sender: TObject);');
11357:     Procedure Sort1IntflistClick( Sender : TObject)');
11358:     Procedure Redo1Click( Sender : TObject)');
11359:     Procedure Tutorial24CleanCode1Click( Sender : TObject)');
11360:     Procedure IndentSelection1Click( Sender : TObject)');
11361:     Procedure UnindentSection1Click( Sender : TObject)');
11362:     Procedure SkyStyle1Click( Sender : TObject)');
11363:     Procedure CountWords1Click( Sender : TObject)');
11364:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)');
11365:     Procedure Memo1GutterClick(Sender : TObject; Button: TMouseButton; X,Y,Line:Integer;
      Mark:TSynEditMark);
11366:     Procedure Bookmark11Click( Sender : TObject)');
11367:     Procedure Bookmark21Click( Sender : TObject)');
11368:     Procedure Bookmark31Click( Sender : TObject)');
11369:     Procedure Bookmark41Click( Sender : TObject)');
11370:     Procedure SynMultiSyn1CustomRange( Sender : TSynMultiSyn;Operation:TRangeOperation; var Range:
      Pointer)');
11371:     'STATMemoryReport', 'boolean', iptrw);
11372:     'IPPort', 'integer', iptrw);
11373:     'COMPort', 'integer', iptrw);
11374:     'lbintflist', 'TListBox', iptrw);
11375:     Function GetStatChange : boolean');
11376:     Procedure SetStatChange( vstat : boolean)');
11377:     Function GetActFileName : string');
11378:     Procedure SetActFileName( vname : string)');
11379:     Function GetLastFileName : string');
11380:     Procedure SetLastFileName( vname : string)');
11381:     Procedure WebScannerDirect( urls : string)');
11382:     Procedure LoadInterfaceList2');
11383:     Function GetStatExecuteShell : boolean');
11384:     Procedure DoEditorExecuteCommand( EditorCommand : word)');
11385:     function GetActiveLineColor: TColor');
11386:     procedure SetActiveLineColor(acolor: TColor)');
11387:     procedure ScriptListbox1Click(Sender: TObject);');
11388:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);');
11389:     procedure EnlargeGutter1Click(Sender: TObject);');
11390:     procedure Tetris1Click(Sender: TObject);
11391:     procedure ToDoList1Click(Sender: TObject);
11392:     procedure ProcessList1Click(Sender: TObject);
11393:     procedure MetricReport1Click(Sender: TObject);
11394:
11395:
11396:
11397: //-------------------------------------------------------------------------
11398: //*************mX4 Editor SynEdit Tools API ********************************
11399: //-------------------------------------------------------------------------
11400: (*-------------------------------------------------------------------------*)
11401: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
11402: begin
11403:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
11404:   with CL.AddClassN(CL.FindClass('TCustomControl'),'TCustomSynEdit') do begin
11405:     Constructor Create( AOwner : TComponent)');
11406:     SelStart', 'Integer', iptrw);
11407:     SelEnd', 'Integer', iptrw);
11408:     AlwaysShowCaret', 'Boolean', iptrw);
11409:     Procedure UpdateCaret');
11410:     Procedure AddKey(Command: TSynEditorCommand; Key1:word; SS1: TShiftState; Key2:word;SS2:TShiftState);
11411:     Procedure AddKey(Command: TSynEditorCommand; Key1:word; SS1: TShiftState; Key2: word;SS2:TShiftState);
11412:     Procedure BeginUndoBlock');
11413:     Procedure BeginUpdate');
11414:     Function CaretInView : Boolean');
11415:     Function CharIndexToRowCol( Index : integer) : TBufferCoord');
11416:     Procedure Clear');
11417:     Procedure ClearAll');
11418:     Procedure ClearBookMark( BookMark : Integer)');
11419:     Procedure ClearSelection');
11420:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)');
11421:     Procedure ClearUndo');
```

```
11422:     Procedure CopyToClipboard');
11423:     Procedure CutToClipboard');
11424:     Procedure DoCopyToClipboard( const SText : string)');
11425:     Procedure EndUndoBlock');
11426:     Procedure EndUpdate');
11427:     Procedure EnsureCursorPosVisible');
11428:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)');
11429:     Procedure FindMatchingBracket');
11430:     Function GetMatchingBracket : TBufferCoord');
11431:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord');
11432:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)');
11433:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean');
11434:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
11435:               : TSynHighlighterAttributes) : boolean');
11436:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
11437:               var TokenType, Start : Integer; var Attri : TSynHighlighterAttributes) : boolean');
11438:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean');
11439:     Function GetWordAtRowCol( const XY : TBufferCoord) : string');
11440:     Procedure GotoBookMark( BookMark : Integer)');
11441:     Procedure GotoLineAndCenter( ALine : Integer)');
11442:     Function IdentChars : TSynIdentChars');
11443:     Procedure InvalidateGutter');
11444:     Procedure InvalidateGutterLine( aLine : integer)');
11445:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)');
11446:     Procedure InvalidateLine( Line : integer)');
11447:     Procedure InvalidateLines( FirstLine, LastLine : integer)');
11448:     Procedure InvalidateSelection');
11449:     Function IsBookmark( BookMark : integer) : boolean');
11450:     Function IsPointInSelection( const Value : TBufferCoord) : boolean');
11451:     Procedure LockUndo');
11452:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord');
11453:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord');
11454:     Function LineToRow( aLine : integer) : integer');
11455:     Function RowToLine( aRow : integer) : integer');
11456:     Function NextWordPos : TBufferCoord');
11457:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord');
11458:     Procedure PasteFromClipboard');
11459:     Function WordStart : TBufferCoord');
11460:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord');
11461:     Function WordEnd : TBufferCoord');
11462:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord');
11463:     Function PrevWordPos : TBufferCoord');
11464:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord');
11465:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord');
11466:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord');
11467:     Procedure Redo');
11468:     Procedure RegisterCommandHandler( const AHandlerProc : THookedCommandEvent; AHandlerData : pointer)');
11469:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint');
11470:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer');
11471:     Function SearchReplace( const ASearch, AReplace : string; AOptions : TSynSearchOptions) : integer');
11472:     Procedure SelectAll');
11473:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)');
11474:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)');
11475:     Procedure SetDefaultKeystrokes');
11476:     Procedure SetSelWord');
11477:     Procedure SetWordBlock( Value : TBufferCoord)');
11478:     Procedure Undo');
11479:     Procedure UnlockUndo');
11480:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)');
11481:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)');
11482:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)');
11483:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)');
11484:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)');
11485:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)');
11486:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)');
11487:     Procedure AddFocusControl( aControl : TWinControl)');
11488:     Procedure RemoveFocusControl( aControl : TWinControl)');
11489:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)');
11490:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)');
11491:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)');
11492:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)');
11493:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)');
11494:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)');
11495:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)');
11496:     Procedure RemoveLinesPointer');
11497:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)');
11498:     Procedure UnHookTextBuffer');
11499:     BlockBegin', 'TBufferCoord', iptrw);
11500:     BlockEnd', 'TBufferCoord', iptrw);
11501:     CanPaste', 'Boolean', iptr);
11502:     CanRedo', 'boolean', iptr);
11503:     CanUndo', 'boolean', iptr);
11504:     CaretX', 'Integer', iptrw);
11505:     CaretY', 'Integer', iptrw);
11506:     CaretXY', 'TBufferCoord', iptrw);
11507:     ActiveLineColor', 'TColor', iptrw);
```

```
11508:     DisplayX', 'Integer', iptr);
11509:     DisplayY', 'Integer', iptr);
11510:     DisplayXY', 'TDisplayCoord', iptr);
11511:     DisplayLineCount', 'integer', iptr);
11512:     CharsInWindow', 'Integer', iptr);
11513:     CharWidth', 'integer', iptr);
11514:     Font', 'TFont', iptrw);
11515:     GutterWidth', 'Integer', iptr);
11516:     Highlighter', 'TSynCustomHighlighter', iptrw);
11517:     LeftChar', 'Integer', iptrw);
11518:     LineHeight', 'integer', iptr);
11519:     LinesInWindow', 'Integer', iptr);
11520:     LineText', 'string', iptrw);
11521:     Lines', 'TStrings', iptrw);
11522:     Marks', 'TSynEditMarkList', iptr);
11523:     MaxScrollWidth', 'integer', iptrw);
11524:     Modified', 'Boolean', iptrw);
11525:     PaintLock', 'Integer', iptr);
11526:     ReadOnly', 'Boolean', iptrw);
11527:     SearchEngine', 'TSynEditSearchCustom', iptrw);
11528:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
11529:     SelTabBlock', 'Boolean', iptr);
11530:     SelTabLine', 'Boolean', iptr);
11531:     SelText', 'string', iptrw);
11532:     StateFlags', 'TSynStateFlags', iptr);
11533:     Text', 'string', iptrw);
11534:     TopLine', 'Integer', iptrw);
11535:     WordAtCursor', 'string', iptr);
11536:     WordAtMouse', 'string', iptr);
11537:     UndoList', 'TSynEditUndoList', iptr);
11538:     RedoList', 'TSynEditUndoList', iptr);
11539:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
11540:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
11541:     BorderStyle', 'TSynBorderStyle', iptrw);
11542:     ExtraLineSpacing', 'integer', iptrw);
11543:     Gutter', 'TSynGutter', iptrw);
11544:     HideSelection', 'boolean', iptrw);
11545:     InsertCaret', 'TSynEditCaretType', iptrw);
11546:     InsertMode', 'boolean', iptrw);
11547:     IsScrolling', 'Boolean', iptr);
11548:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
11549:     MaxUndo', 'Integer', iptrw);
11550:     Options', 'TSynEditorOptions', iptrw);
11551:     OverwriteCaret', 'TSynEditCaretType', iptrw);
11552:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
11553:     ScrollHintColor', 'TColor', iptrw);
11554:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
11555:     ScrollBars', 'TScrollStyle', iptrw);
11556:     SelectedColor', 'TSynSelectedColor', iptrw);
11557:     SelectionMode', 'TSynSelectionMode', iptrw);
11558:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
11559:     TabWidth', 'integer', iptrw);     WantReturns', 'boolean', iptrw);
11560:     WantTabs', 'boolean', iptrw);     WordWrap', 'boolean', iptrw);
11561:     WordWrapGlyph', 'TSynGlyph', iptrw);
11562:     OnChange', 'TNotifyEvent', iptrw);
11563:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
11564:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
11565:     OnContextHelp', 'TContextHelpEvent', iptrw);
11566:     OnDropFiles', 'TDropFilesEvent', iptrw);
11567:     OnGutterClick', 'TGutterClickEvent', iptrw);
11568:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
11569:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
11570:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
11571:     OnPaint', 'TPaintEvent', iptrw);
11572:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
11573:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
11574:     OnReplaceText', 'TReplaceTextEvent', iptrw);
11575:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
11576:     OnStatusChange', 'TStatusChangeEvent', iptrw);
11577:     OnPaintTransient', 'TPaintTransient', iptrw);
11578:     OnScroll', 'TScrollEvent', iptrw);
11579:   end;
11580:   end;
11581:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)');
11582:  Function GetPlaceableHighlighters : TSynHighlighterList');
11583:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string');
11584:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string');
11585:  Procedure GetEditorCommandValues( Proc : TGetStrProc)');
11586:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)');
11587:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean');
11588:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean');
11589:  Function ConvertCodeStringToExtended( AString : String) : String');
11590:  Function ConvertExtendedToCodeString( AString : String) : String');
11591:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand');
11592:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand');
11593:  Function IndexToEditorCommand( const AIndex : Integer) : Integer');
```

```
11594:
11595:    TSynEditorOption = (
11596:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
11597:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
11598:                                // preceding line
11599:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
11600:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
11601:                                //direction any more
11602:     eoDragDropEditing,         //Allows to select a block of text and drag it within document to another
11603:                                // location
11604:     eoDropFiles,               //Allows the editor accept OLE file drops
11605:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
11606:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
11607:     eoGroupUndo,               //When undoing/redoing actions, handle all continous changes the same kind
11608:                                // in one call
11609:                                //instead undoing/redoing each command separately
11610:     eoHalfPageScroll,          //When scrolling with page-up and page-down commands, only scroll a half
11611:                                //page at a time
11612:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
11613:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
11614:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
11615:     eoNoCaret,                 //Makes it so the caret is never visible
11616:     eoNoSelection,             //Disables selecting text
11617:     eoRightMouseMovesCursor,   //When clicking with the right mouse for a popup menu, move the cursor to
11618:                                //that location
11619:     eoScrollByOneLess,         //Forces scrolling to be one less
11620:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
11621:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
11622:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
11623:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
11624:     eoShowSpecialChars,        //Shows the special Characters
11625:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
11626:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
11627:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
11628:     eoTabIndent,               //When active <Tab> and <Shift><Tab> act as block indent, unindent when text
11629:                                // is selected
11630:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
11631:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
11632:
11633:     *********************************Important Editor Short Cuts*********************************);
11634:     Double click to select a word and count words with highlightning.
11635:     Triple click to select a line.
11636:     CTRL+SHIFT+click to extend a selection.
11637:     Drag with the ALT key down to select columns of text !!!
11638:     Drag and drop is supported.
11639:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
11640:     Type CTRL+A to select all.
11641:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
11642:     Type Home to position cursor at beginning of current line and End to position it at end of line.
11643:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
11644:     Page Up and Page Down work as expected.
11645:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
11646:
11647:     http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
11648:  var
11649:   ReservedWords: array[0..75] of string =
11650:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
11651:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
11652:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
11653:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
11654:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
11655:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
11656:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
11657:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
11658:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
11659:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
11660:      'public', 'published');
11661:    AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',''); t,t1,t2,t3: boolean;
11662:
11663: //-----------------------------------------------------------------------------
11664: //**************End of mX4 Public  Tools API *********************************
11665: //-----------------------------------------------------------------------------
11666:
11667: Amount of Functions: 6826
11668: Amount of Procedures: 4059
11669: Amount of Constructors: 661
11670: Totals of Calls: 11546
11671: SHA1: 76ECD732E1531108B35457D56282B0BBBBB20BAA
11672:
11673: ************************************************************
11674:  Short Manual with 25 Tips!
11675: ************************************************************
11676: - Install: just save your maxboxdef.ini before and then extract the zip file!
11677:
11678: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
11679:
```

```
11680: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
11681: - Menu: With <Crtl><F3> you can search for code on examples
11682: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
11683: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
11684: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
11685:
11686: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
11687:
11688: - Context Menu: You can printout your scripts as a pdf-file or html-export
11689: - Context: You do have a context menu with the right mouse click
11690:
11691: - Menu: With the UseCase Editor you can convert graphic formats too.
11692: - Menu: On menu Options you find Addons as compiled scripts
11693:
11694: - IDE: You don't need a mouse to handle maXbox, use shortcuts
11695: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
11696: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
11697: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
11698:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
11699:
11700: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
11701: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
11702: - Code: If you code a loop till key-pressed use function: isKeyPressed;
11703: - Code: Macro set the macros #name, #date, #host, #path, #file, #head #sign, see Tutorial
      maxbox_starter25.pdf
11704:
11705: - Editor: - Dbl Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
11706:          to delete and Click and mark to drag a bookmark
11707:
11708: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
11709: - IDE: A file info with system and script information you find in menu Program/Information
11710: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
11711: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
11712: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
11713:
11714:
11715:
11716:
11717:
11718: ************************************************************
11719:  unit List asm internal end
11720: ************************************************************
11721: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
11722: 02 unit SIRegister_IdStrings                   //Indy Sockets
11723: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
11724: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
11725: 05 unit IFSI_WinForm1puzzle;                   //maXbox
11726: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
11727: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
11728: 08 unit RIRegister_MathMax_Routines(exec);    //Jedi & Delphi
11729: 09 unit RIRegister_IdGlobal_Routines(exec);   //Indy Sockets
11730: 10 unit RIRegister_SysUtils_Routines(Exec);   //Delphi
11731: 11 unit uPSI_IdTCPConnection;                 //Indy some functions
11732: 12 unit uPSCompiler.pas;                      //PS kernel functions
11733: 13 unit uPSI_DBCommon;                        //DB Common_Routines and Types
11734: 14 unit uPSI_Printers.pas                     //Delphi VCL
11735: 15 unit uPSI_MPlayer.pas                     //Delphi VCL
11736: 16 unit uPSC_comobj;                         //COM Functions
11737: 17 unit uPSI_Clipbrd;                        //Delphi VCL
11738: 18 unit Filectrl in IFSI_SysUtils_max;      //VCL Runtime
11739: 19 unit uPSI_SqlExpr;                       //DBX3
11740: 20 unit uPSI_ADODB;                         //ADODB
11741: 21 unit uPSI_StrHlpr;                       //String Helper Routines
11742: 22 unit uPSI_DateUtils;                     //Expansion to DateTimeLib
11743: 23 unit uPSI_FileUtils;                     //Expansion to Sys/File Utils
11744: 24 unit JUtils / gsUtils;                   //Jedi / Metabase
11745: 25 unit JvFunctions_max;                    //Jedi Functions
11746: 26 unit HTTPParser;                         //Delphi VCL
11747: 27 unit HTTPUtil;                           //Delphi VCL
11748: 28 unit uPSI_XMLUtil;                       //Delphi VCL
11749: 29 unit uPSI_SOAPHTTPClient;                //Delphi VCL SOAP WebService V3.5
11750: 30 unit uPSI_Contnrs;                       //Delphi RTL Container of Classes
11751: 31 unit uPSI_MaskUtils;                     //RTL Edit and Mask functions
11752: 32 unit uPSI_MyBigInt;                      //big integer class with Math
11753: 33 unit uPSI_ConvUtils;                     //Delphi VCL Conversions engine
11754: 34 unit Types_Variants;                     //Delphi\Win32\rtl\sys
11755: 35 unit uPSI_IdHashSHA1;                    //Indy Crypto Lib
11756: 36 unit uPSI_IdHashMessageDigest           //Indy Crypto;
11757: 37 unit uPSI_IdASN1Util;                    //Indy ASN1Utility Routines;
11758: 38 unit uPSI_IdLogFile;                     //Indy Logger from LogBase
11759: 39 unit uPSI_IdIcmpClient;                  //Indy Ping ICMP
11760: 40 unit uPSI_IdHashMessageDigest_max       //Indy Crypto &OpenSSL;
11761: 41 unit uPSI_FileCtrl;                      //Delphi RTL
11762: 42 unit uPSI_Outline;                       //Delphi VCL
11763: 43 unit uPSI_ScktComp;                      //Delphi RTL
11764: 44 unit uPSI_Calendar;                      //Delphi VCL
```

```
11765: 45 unit uPSI_VListView                    //VListView;
11766: 46 unit uPSI_DBGrids;                     //Delphi VCL
11767: 47 unit uPSI_DBCtrls;                     //Delphi VCL
11768: 48 unit ide_debugoutput;                  //maXbox
11769: 49 unit uPSI_ComCtrls;                    //Delphi VCL
11770: 50 unit uPSC_stdctrls+;                   //Delphi VCL
11771: 51 unit uPSI_Dialogs;                     //Delphi VCL
11772: 52 unit uPSI_StdConvs;                   //Delphi RTL
11773: 53 unit uPSI_DBClient;                   //Delphi RTL
11774: 54 unit uPSI_DBPlatform;                 //Delphi RTL
11775: 55 unit uPSI_Provider;                   //Delphi RTL
11776: 56 unit uPSI_FMTBcd;                     //Delphi RTL
11777: 57 unit uPSI_DBCGrids;                   //Delphi VCL
11778: 58 unit uPSI_CDSUtil;                    //MIDAS
11779: 59 unit uPSI_VarHlpr;                    //Delphi RTL
11780: 60 unit uPSI_ExtDlgs;                    //Delphi VCL
11781: 61 unit sdpStopwatch;                    //maXbox
11782: 62 unit uPSI_JclStatistics;              //JCL
11783: 63 unit uPSI_JclLogic;                   //JCL
11784: 64 unit uPSI_JclMiscel;                  //JCL
11785: 65 unit uPSI_JclMath_max;                //JCL RTL
11786: 66 unit uPSI_uTPLb_StreamUtils;          //LockBox 3
11787: 67 unit uPSI_MathUtils;                  //BCB
11788: 68 unit uPSI_JclMultimedia;              //JCL
11789: 69 unit uPSI_WideStrUtils;               //Delphi API/RTL
11790: 70 unit uPSI_GraphUtil;                  //Delphi RTL
11791: 71 unit uPSI_TypeTrans;                  //Delphi RTL
11792: 72 unit uPSI_HTTPApp;                    //Delphi VCL
11793: 73 unit uPSI_DBWeb;                      //Delphi VCL
11794: 74 unit uPSI_DBBdeWeb;                   //Delphi VCL
11795: 75 unit uPSI_DBXpressWeb;                //Delphi VCL
11796: 76 unit uPSI_ShadowWnd;                  //Delphi VCL
11797: 77 unit uPSI_ToolWin;                    //Delphi VCL
11798: 78 unit uPSI_Tabs;                       //Delphi VCL
11799: 79 unit uPSI_JclGraphUtils;              //JCL
11800: 80 unit uPSI_JclCounter;                 //JCL
11801: 81 unit uPSI_JclSysInfo;                 //JCL
11802: 82 unit uPSI_JclSecurity;                //JCL
11803: 83 unit uPSI_JclFileUtils;               //JCL
11804: 84 unit uPSI_IdUserAccounts;             //Indy
11805: 85 unit uPSI_IdAuthentication;           //Indy
11806: 86 unit uPSI_uTPLb_AES;                  //LockBox 3
11807: 87 unit uPSI_IdHashSHA1;                 //LockBox 3
11808: 88 unit uTPLb_BlockCipher;               //LockBox 3
11809: 89 unit uPSI_ValEdit.pas;                //Delphi VCL
11810: 90 unit uPSI_JvVCLUtils;                 //JCL
11811: 91 unit uPSI_JvDBUtil;                   //JCL
11812: 92 unit uPSI_JvDBUtils;                  //JCL
11813: 93 unit uPSI_JvAppUtils;                 //JCL
11814: 94 unit uPSI_JvCtrlUtils;                //JCL
11815: 95 unit uPSI_JvFormToHtml;               //JCL
11816: 96 unit uPSI_JvParsing;                  //JCL
11817: 97 unit uPSI_SerDlgs;                    //Toolbox
11818: 98 unit uPSI_Serial;                     //Toolbox
11819: 99 unit uPSI_JvComponent;                //JCL
11820: 100 unit uPSI_JvCalc;                    //JCL
11821: 101 unit uPSI_JvBdeUtils;                //JCL
11822: 102 unit uPSI_JvDateUtil;                //JCL
11823: 103 unit uPSI_JvGenetic;                 //JCL
11824: 104 unit uPSI_JclBase;                   //JCL
11825: 105 unit uPSI_JvUtils;                   //JCL
11826: 106 unit uPSI_JvStrUtil;                 //JCL
11827: 107 unit uPSI_JvStrUtils;                //JCL
11828: 108 unit uPSI_JvFileUtil;                //JCL
11829: 109 unit uPSI_JvMemoryInfos;             //JCL
11830: 110 unit uPSI_JvComputerInfo;            //JCL
11831: 111 unit uPSI_JvgCommClasses;            //JCL
11832: 112 unit uPSI_JvgLogics;                 //JCL
11833: 113 unit uPSI_JvLED;                     //JCL
11834: 114 unit uPSI_JvTurtle;                  //JCL
11835: 115 unit uPSI_SortThds; unit uPSI_ThSort;  //maXbox
11836: 116 unit uPSI_JvgUtils;                  //JCL
11837: 117 unit uPSI_JvExprParser;              //JCL
11838: 118 unit uPSI_HexDump;                   //Borland
11839: 119 unit uPSI_DBLogDlg;                  //VCL
11840: 120 unit uPSI_SqlTimSt;                  //RTL
11841: 121 unit uPSI_JvHtmlParser;              //JCL
11842: 122 unit uPSI_JvgXMLSerializer;          //JCL
11843: 123 unit uPSI_JvJCLUtils;                //JCL
11844: 124 unit uPSI_JvStrings;                 //JCL
11845: 125 unit uPSI_uTPLb_IntegerUtils;        //TurboPower
11846: 126 unit uPSI_uTPLb_HugeCardinal;        //TurboPower
11847: 127 unit uPSI_uTPLb_HugeCardinalUtils;   //TurboPower
11848: 128 unit uPSI_SynRegExpr;                //SynEdit
11849: 129 unit uPSI_StUtils;                   //SysTools4
11850: 130 unit uPSI_StToHTML;                  //SysTools4
```

```
11851: 131 unit uPSI_StStrms;                        //SysTools4
11852: 132 unit uPSI_StFIN;                          //SysTools4
11853: 133 unit uPSI_StAstroP;                       //SysTools4
11854: 134 unit uPSI_StStat;                         //SysTools4
11855: 135 unit uPSI_StNetCon;                       //SysTools4
11856: 136 unit uPSI_StDecMth;                       //SysTools4
11857: 137 unit uPSI_StOStr;                         //SysTools4
11858: 138 unit uPSI_StPtrns;                        //SysTools4
11859: 139 unit uPSI_StNetMsg;                       //SysTools4
11860: 140 unit uPSI_StMath;                         //SysTools4
11861: 141 unit uPSI_StExpEng;                       //SysTools4
11862: 142 unit uPSI_StCRC;                          //SysTools4
11863: 143 unit uPSI_StExport,                       //SysTools4
11864: 144 unit uPSI_StExpLog,                       //SysTools4
11865: 145 unit uPSI_ActnList;                       //Delphi VCL
11866: 146 unit uPSI_jpeg;                           //Borland
11867: 147 unit uPSI_StRandom;                       //SysTools4
11868: 148 unit uPSI_StDict;                         //SysTools4
11869: 149 unit uPSI_StBCD;                          //SysTools4
11870: 150 unit uPSI_StTxtDat;                       //SysTools4
11871: 151 unit uPSI_StRegEx;                        //SysTools4
11872: 152 unit uPSI_IMouse;                         //VCL
11873: 153 unit uPSI_SyncObjs;                       //VCL
11874: 154 unit uPSI_AsyncCalls;                     //Hausladen
11875: 155 unit uPSI_ParallelJobs;                   //Saraiva
11876: 156 unit uPSI_Variants;                       //VCL
11877: 157 unit uPSI_VarCmplx;                       //VCL Wolfram
11878: 158 unit uPSI_DTDSchema;                      //VCL
11879: 159 unit uPSI_ShLwApi;                        //Brakel
11880: 160 unit uPSI_IBUtils;                        //VCL
11881: 161 unit uPSI_CheckLst;                       //VCL
11882: 162 unit uPSI_JvSimpleXml;                    //JCL
11883: 163 unit uPSI_JclSimpleXml;                   //JCL
11884: 164 unit uPSI_JvXmlDatabase;                  //JCL
11885: 165 unit uPSI_JvMaxPixel;                     //JCL
11886: 166 unit uPSI_JvItemsSearchs;                 //JCL
11887: 167 unit uPSI_StExpEng2;                      //SysTools4
11888: 168 unit uPSI_StGenLog;                       //SysTools4
11889: 169 unit uPSI_JvLogFile;                      //Jcl
11890: 170 unit uPSI_CPort;                          //ComPort Lib v4.11
11891: 171 unit uPSI_CPortCtl;                       //ComPort
11892: 172 unit uPSI_CPortEsc;                       //ComPort
11893: 173 unit BarCodeScaner;                       //ComPort
11894: 174 unit uPSI_JvGraph;                        //JCL
11895: 175 unit uPSI_JvComCtrls;                     //JCL
11896: 176 unit uPSI_GUITesting;                     //D Unit
11897: 177 unit uPSI_JvFindFiles;                    //JCL
11898: 178 unit uPSI_StSystem;                       //SysTools4
11899: 179 unit uPSI_JvKeyboardStates;               //JCL
11900: 180 unit uPSI_JvMail;                         //JCL
11901: 181 unit uPSI_JclConsole;                     //JCL
11902: 182 unit uPSI_JclLANMan;                      //JCL
11903: 183 unit uPSI_IdCustomHTTPServer;             //Indy
11904: 184 unit IdHTTPServer                         //Indy
11905: 185 unit uPSI_IdTCPServer;                    //Indy
11906: 186 unit uPSI_IdSocketHandle;                 //Indy
11907: 187 unit uPSI_IdIOHandlerSocket;              //Indy
11908: 188 unit IdIOHandler;                         //Indy
11909: 189 unit uPSI_cutils;                         //Bloodshed
11910: 190 unit uPSI_BoldUtils;                      //boldsoft
11911: 191 unit uPSI_IdSimpleServer;                 //Indy
11912: 192 unit uPSI_IdSSLOpenSSL;                   //Indy
11913: 193 unit uPSI_IdMultipartFormData;            //Indy
11914: 194 unit uPSI_SynURIOpener;                   //SynEdit
11915: 195 unit uPSI_PerlRegEx;                      //PCRE
11916: 196 unit uPSI_IdHeaderList;                   //Indy
11917: 197 unit uPSI_StFirst;                        //SysTools4
11918: 198 unit uPSI_JvCtrls;                        //JCL
11919: 199 unit uPSI_IdTrivialFTPBase;               //Indy
11920: 200 unit uPSI_IdTrivialFTP;                   //Indy
11921: 201 unit uPSI_IdUDPBase;                      //Indy
11922: 202 unit uPSI_IdUDPClient;                    //Indy
11923: 203 unit uPSI_utypes;                         //for DMath.DLL
11924: 204 unit uPSI_ShellAPI;                       //Borland
11925: 205 unit uPSI_IdRemoteCMDClient;              //Indy
11926: 206 unit uPSI_IdRemoteCMDServer;              //Indy
11927: 207 unit IdRexecServer;                       //Indy
11928: 208 unit IdRexec; (unit uPSI_IdRexec;)        //Indy
11929: 209 unit IdUDPServer;                         //Indy
11930: 210 unit IdTimeUDPServer;                     //Indy
11931: 211 unit IdTimeServer;                        //Indy
11932: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)  //Indy
11933: 213 unit uPSI_IdIPWatch;                      //Indy
11934: 214 unit uPSI_IdIrcServer;                    //Indy
11935: 215 unit uPSI_IdMessageCollection;            //Indy
11936: 216 unit uPSI_cPEM;                           //Fundamentals 4
```

```
11937: 217 unit uPSI_cFundamentUtils;                //Fundamentals 4
11938: 218 unit uPSI_uwinplot;                       //DMath
11939: 219 unit uPSI_xrtl_util_CPUUtils;             //ExtentedRTL
11940: 220 unit uPSI_GR32_System;                    //Graphics32
11941: 221 unit uPSI_cFileUtils;                     //Fundamentals 4
11942: 222 unit uPSI_cDateTime; (timemachine)        //Fundamentals 4
11943: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
11944: 224 unit uPSI_cRandom;                        //Fundamentals 4
11945: 225 unit uPSI_ueval;                          //DMath
11946: 226 unit uPSI_xrtl_net_URIUtils;              //ExtendedRTL
11947: 227 unit xrtl_net_URIUtils;                   //ExtendedRTL
11948: 228 unit uPSI_ufft;  (FFT)                    //DMath
11949: 229 unit uPSI_DBXChannel;                     //Delphi
11950: 230 unit uPSI_DBXIndyChannel;                 //Delphi Indy
11951: 231 unit uPSI_xrtl_util_COMCat;               //ExtendedRTL
11952: 232 unit uPSI_xrtl_util_StrUtils;             //ExtendedRTL
11953: 233 unit uPSI_xrtl_util_VariantUtils;         //ExtendedRTL
11954: 234 unit uPSI_xrtl_util_FileUtils;            //ExtendedRTL
11955: 235 unit xrtl_util_Compat;                    //ExtendedRTL
11956: 236 unit uPSI_OleAuto;                        //Borland
11957: 237 unit uPSI_xrtl_util_COMUtils;             //ExtendedRTL
11958: 238 unit uPSI_CmAdmCtl;                       //Borland
11959: 239 unit uPSI_ValEdit2;                       //VCL
11960: 240 unit uPSI_GR32;  //Graphics32            //Graphics32
11961: 241 unit uPSI_GR32_Image;                     //Graphics32
11962: 242 unit uPSI_xrtl_util_TimeUtils;            //ExtendedRTL
11963: 243 unit uPSI_xrtl_util_TimeZone;             //ExtendedRTL
11964: 244 unit uPSI_xrtl_util_TimeStamp;            //ExtendedRTL
11965: 245 unit uPSI_xrtl_util_Map;                  //ExtendedRTL
11966: 246 unit uPSI_xrtl_util_Set;                  //ExtendedRTL
11967: 247 unit uPSI_CPortMonitor;                   //ComPort
11968: 248 unit uPSI_StIniStm;                       //SysTools4
11969: 249 unit uPSI_GR32_ExtImage;                  //Graphics32
11970: 250 unit uPSI_GR32_OrdinalMaps;               //Graphics32
11971: 251 unit uPSI_GR32_Rasterizers;               //Graphics32
11972: 252 unit uPSI_xrtl_util_Exception;            //ExtendedRTL
11973: 253 unit uPSI_xrtl_util_Value;                //ExtendedRTL
11974: 254 unit uPSI_xrtl_util_Compare;              //ExtendedRTL
11975: 255 unit uPSI_FlatSB;                         //VCL
11976: 256 unit uPSI_JvAnalogClock;                  //JCL
11977: 257 unit uPSI_JvAlarms;                       //JCL
11978: 258 unit uPSI_JvSQLS;                         //JCL
11979: 259 unit uPSI_JvDBSecur;                      //JCL
11980: 260 unit uPSI_JvDBQBE;                        //JCL
11981: 261 unit uPSI_JvStarfield;                    //JCL
11982: 262 unit uPSI_JVCLMiscal;                     //JCL
11983: 263 unit uPSI_JvProfiler32;                   //JCL
11984: 264 unit uPSI_JvDirectories,                  //JCL
11985: 265 unit uPSI_JclSchedule,                    //JCL
11986: 266 unit uPSI_JclSvcCtrl,                     //JCL
11987: 267 unit uPSI_JvSoundControl,                 //JCL
11988: 268 unit uPSI_JvBDESQLScript,                 //JCL
11989: 269 unit uPSI_JvgDigits,                      //JCL>
11990: 270 unit uPSI_ImgList;                        //TCustomImageList
11991: 271 unit uPSI_JclMIDI;                        //JCL>
11992: 272 unit uPSI_JclWinMidi;                     //JCL>
11993: 273 unit uPSI_JclNTFS;                        //JCL>
11994: 274 unit uPSI_JclAppInst;                     //JCL>
11995: 275 unit uPSI_JvRle;                          //JCL>
11996: 276 unit uPSI_JvRas32;                        //JCL>
11997: 277 unit uPSI_JvImageDrawThread,              //JCL>
11998: 278 unit uPSI_JvImageWindow,                  //JCL>
11999: 279 unit uPSI_JvTransparentForm;              //JCL>
12000: 280 unit uPSI_JvWinDialogs;                   //JCL>
12001: 281 unit uPSI_JvSimLogic,                     //JCL>
12002: 282 unit uPSI_JvSimIndicator,                 //JCL>
12003: 283 unit uPSI_JvSimPID,                       //JCL>
12004: 284 unit uPSI_JvSimPIDLinker,                 //JCL>
12005: 285 unit uPSI_IdRFCReply;                     //Indy
12006: 286 unit uPSI_IdIdent;                        //Indy
12007: 287 unit uPSI_IdIdentServer;                  //Indy
12008: 288 unit uPSI_JvPatchFile;                    //JCL
12009: 289 unit uPSI_StNetPfm;                       //SysTools4
12010: 290 unit uPSI_StNet;                          //SysTools4
12011: 291 unit uPSI_JclPeImage;                     //JCL
12012: 292 unit uPSI_JclPrint;                       //JCL
12013: 293 unit uPSI_JclMime;                        //JCL
12014: 294 unit uPSI_JvRichEdit;                     //JCL
12015: 295 unit uPSI_JvDBRichEd;                     //JCL
12016: 296 unit uPSI_JvDice;                         //JCL
12017: 297 unit uPSI_JvFloatEdit;                    //JCL 3.9.8
12018: 298 unit uPSI_JvDirFrm;                       //JCL
12019: 299 unit uPSI_JvDualList;                     //JCL
12020: 300 unit uPSI_JvSwitch;                       ////JCL
12021: 301 unit uPSI_JvTimerLst;                     ////JCL
12022: 302 unit uPSI_JvMemTable;                     //JCL
```

```
12023: 303 unit uPSI_JvObjStr;                    //JCL
12024: 304 unit uPSI_StLArr;                     //SysTools4
12025: 305 unit uPSI_StWmDCpy;                   //SysTools4
12026: 306 unit uPSI_StText;                     //SysTools4
12027: 307 unit uPSI_StNTLog;                    //SysTools4
12028: 308 unit uPSI_xrtl_math_Integer;          //ExtendedRTL
12029: 309 unit uPSI_JvImagPrvw;                 //JCL
12030: 310 unit uPSI_JvFormPatch;                //JCL
12031: 311 unit uPSI_JvPicClip;                  //JCL
12032: 312 unit uPSI_JvDataConv;                 //JCL
12033: 313 unit uPSI_JvCpuUsage;                 //JCL
12034: 314 unit uPSI_JclUnitConv_mX2;            //JCL
12035: 315 unit JvDualListForm;                  //JCL
12036: 316 unit uPSI_JvCpuUsage2;                //JCL
12037: 317 unit uPSI_JvParserForm;               //JCL
12038: 318 unit uPSI_JvJanTreeView;              //JCL
12039: 319 unit uPSI_JvTransLED;                 //JCL
12040: 320 unit uPSI_JvPlaylist;                 //JCL
12041: 321 unit uPSI_JvFormAutoSize;             //JCL
12042: 322 unit uPSI_JvYearGridEditForm;         //JCL
12043: 323 unit uPSI_JvMarkupCommon;             //JCL
12044: 324 unit uPSI_JvChart;                    //JCL
12045: 325 unit uPSI_JvXPCore;                   //JCL
12046: 326 unit uPSI_JvXPCoreUtils;              //JCL
12047: 327 unit uPSI_StatsClasses;               //mX4
12048: 328 unit uPSI_ExtCtrls2;                  //VCL
12049: 329 unit uPSI_JvUrlGrabbers;              //JCL
12050: 330 unit uPSI_JvXmlTree;                  //JCL
12051: 331 unit uPSI_JvWavePlayer;               //JCL
12052: 332 unit uPSI_JvUnicodeCanvas;            //JCL
12053: 333 unit uPSI_JvTFUtils;                  //JCL
12054: 334 unit uPSI_IdServerIOHandler;          //Indy
12055: 335 unit uPSI_IdServerIOHandlerSocket;    //Indy
12056: 336 unit uPSI_IdMessageCoder;             //Indy
12057: 337 unit uPSI_IdMessageCoderMIME;         //Indy
12058: 338 unit uPSI_IdMIMETypes;                //Indy
12059: 339 unit uPSI_JvConverter;                //JCL
12060: 340 unit uPSI_JvCsvParse;                 //JCL
12061: 341 unit uPSI_umath;  unit uPSI_ugamma;   //DMath
12062: 342 unit uPSI_ExcelExport;(Native:TJsExcelExport)
12063: 343 unit uPSI_JvDBGridExport;             //JCL
12064: 344 unit uPSI_JvgExport;                  //JCL
12065: 345 unit uPSI_JvSerialMaker;              //JCL
12066: 346 unit uPSI_JvWin32;                    //JCL
12067: 347 unit uPSI_JvPaintFX;                  //JCL
12068: 348 unit uPSI_JvOracleDataSet; (beta)     //JCL
12069: 349 unit uPSI_JvValidators; (preview)     //JCL
12070: 350 unit uPSI_JvNTEventLog;               //JCL
12071: 351 unit uPSI_ShellZipTool;               //mX4
12072: 352 unit uPSI_JvJoystick;                 //JCL
12073: 353 unit uPSI_JvMailSlots;                //JCL
12074: 354 unit uPSI_JclComplex;                 //JCL
12075: 355 unit uPSI_SynPdf;                     //Synopse
12076: 356 unit uPSI_Registry;                   //VCL
12077: 357 unit uPSI_TlHelp32;                   //VCL
12078: 358 unit uPSI_JclRegistry;                //JCL
12079: 359 unit uPSI_JvAirBrush;                 //JCL
12080: 360 unit uPSI_mORMotReport;               //Synopse
12081: 361 unit uPSI_JclLocales;                 //JCL
12082: 362 unit uPSI_SynEdit;                    //SynEdit
12083: 363 unit uPSI_SynEditTypes;               //SynEdit
12084: 364 unit uPSI_SynMacroRecorder;           //SynEdit
12085: 365 unit uPSI_LongIntList;                //SynEdit
12086: 366 unit uPSI_devcutils;                  //DevC
12087: 367 unit uPSI_SynEditMiscClasses;         //SynEdit
12088: 368 unit uPSI_SynEditRegexSearch;         //SynEdit
12089: 369 unit uPSI_SynEditHighlighter;         //SynEdit
12090: 370 unit uPSI_SynHighlighterPas;          //SynEdit
12091: 371 unit uPSI_JvSearchFiles;              //JCL
12092: 372 unit uPSI_SynHighlighterAny;          //Lazarus
12093: 373 unit uPSI_SynEditKeyCmds;             //SynEdit
12094: 374 unit uPSI_SynEditMiscProcs,           //SynEdit
12095: 375 unit uPSI_SynEditKbdHandler           //SynEdit
12096: 376 unit uPSI_JvAppInst,                  //JCL
12097: 377 unit uPSI_JvAppEvent;                 //JCL
12098: 378 unit uPSI_JvAppCommand;               //JCL
12099: 379 unit uPSI_JvAnimTitle;                //JCL
12100: 380 unit uPSI_JvAnimatedImage;            //JCL
12101: 381 unit uPSI_SynEditExport;              //SynEdit
12102: 382 unit uPSI_SynExportHTML;              //SynEdit
12103: 383 unit uPSI_SynExportRTF;               //SynEdit
12104: 384 unit uPSI_SynEditSearch;              //SynEdit
12105: 385 unit uPSI_fMain_back                  //maXbox;
12106: 386 unit uPSI_JvZoom;                     //JCL
12107: 387 unit uPSI_PMrand;                     //PM
12108: 388 unit uPSI_JvSticker;                  //JCL
```

```
12109: 389 unit uPSI_XmlVerySimple;                  //mX4
12110: 390 unit uPSI_Services;                       //ExtPascal
12111: 391 unit uPSI_ExtPascalUtils;                 //ExtPascal
12112: 392 unit uPSI_SocketsDelphi;                  //ExtPascal
12113: 393 unit uPSI_StBarC;                        //SysTools
12114: 394 unit uPSI_StDbBarC;                      //SysTools
12115: 395 unit uPSI_StBarPN;                       //SysTools
12116: 396 unit uPSI_StDbPNBC;                      //SysTools
12117: 397 unit uPSI_StDb2DBC;                      //SysTools
12118: 398 unit uPSI_StMoney;                       //SysTools
12119: 399 unit uPSI_JvForth;                       //JCL
12120: 400 unit uPSI_RestRequest;                   //mX4
12121: 401 unit uPSI_HttpRESTConnectionIndy;        //mX4
12122: 402 unit uPSI_JvXmlDatabase; //update       //JCL
12123: 403 unit uPSI_StAstro;                       //SysTools
12124: 404 unit uPSI_StSort;                        //SysTools
12125: 405 unit uPSI_StDate;                        //SysTools
12126: 406 unit uPSI_StDateSt;                      //SysTools
12127: 407 unit uPSI_StBase;                        //SysTools
12128: 408 unit uPSI_StVInfo;                       //SysTools
12129: 409 unit uPSI_JvBrowseFolder;                //JCL
12130: 410 unit uPSI_JvBoxProcs;                    //JCL
12131: 411 unit uPSI_urandom; (unit uranuvag;)      //DMath
12132: 412 unit uPSI_usimann; (unit ugenalg;)       //DMath
12133: 413 unit uPSI_JvHighlighter;                 //JCL
12134: 414 unit uPSI_Diff;                          //mX4
12135: 415 unit uPSI_SpringWinAPI;                  //DSpring
12136: 416 unit uPSI_StBits;                        //SysTools
12137: 417 unit uPSI_TomDBQue;                      //mX4
12138: 418 unit uPSI_MultilangTranslator;           //mX4
12139: 419 unit uPSI_HyperLabel;                    //mX4
12140: 420 unit uPSI_Starter;                       //mX4
12141: 421 unit uPSI_FileAssocs;                    //devC
12142: 422 unit uPSI_devFileMonitorX;               //devC
12143: 423 unit uPSI_devrun;                        //devC
12144: 424 unit uPSI_devExec;                       //devC
12145: 425 unit uPSI_oysUtils;                      //devC
12146: 426 unit uPSI_DosCommand;                    //devC
12147: 427 unit uPSI_CppTokenizer;                  //devC
12148: 428 unit uPSI_JvHLParser;                    //devC
12149: 429 unit uPSI_JclMapi;                       //JCL
12150: 430 unit uPSI_JclShell;                      //JCL
12151:
12152:
12153:
12154:
12155: All maXbox Tutorials Table of Content
12156: ////////////////////////////////////////////////////////////////////////////////
12157:
12158:     Tutorial 00 Function-Coding  (Blix the Programmer)
12159:     Tutorial 01 Procedural-Coding
12160:     Tutorial 02 OO-Programming
12161:     Tutorial 03 Modular Coding
12162:     Tutorial 04 UML Use Case Coding
12163:     Tutorial 05 Internet Coding
12164:     Tutorial 06 Network Coding
12165:     Tutorial 07 Game Graphics Coding
12166:     Tutorial 08 Operating System Coding
12167:     Tutorial 09 Database Coding
12168:     Tutorial 10 Statistic Coding
12169:     Tutorial 11 Forms Coding
12170:     Tutorial 12 SQL DB Coding
12171:     Tutorial 13 Crypto Coding
12172:     Tutorial 14 Parallel Coding
12173:     Tutorial 15 Serial RS232 Coding
12174:     Tutorial 16 Event Driven Coding
12175:     Tutorial 17 Web Server Coding
12176:     Tutorial 18 Arduino System Coding
12177:     Tutorial 19 WinCOM /Arduino Coding
12178:     Tutorial 20 Regular Expressions RegEx
12179:     Tutorial 21 Android Coding (coming 2013)
12180:     Tutorial 22 Services Programming
12181:     Tutorial 23 Real Time Systems (coming 2013)
12182:     Tutorial 24 Clean Code (coming 2014)
12183:     Tutorial 25 maXbox Configuration I+II
12184:     Tutorial 26 XML & TreeView (coming 2013)
12185:
12186:
12187:
12188: http://sourceforge.net/projects/maXbox   #locs:0
12189: http://sourceforge.net/apps/mediawiki/maXbox
12190:   ---- code_cleared_checked----
```